Consultative Committee for Space Data Systems

RECOMMENDATION FOR SPACE DATA SYSTEM STANDARDS

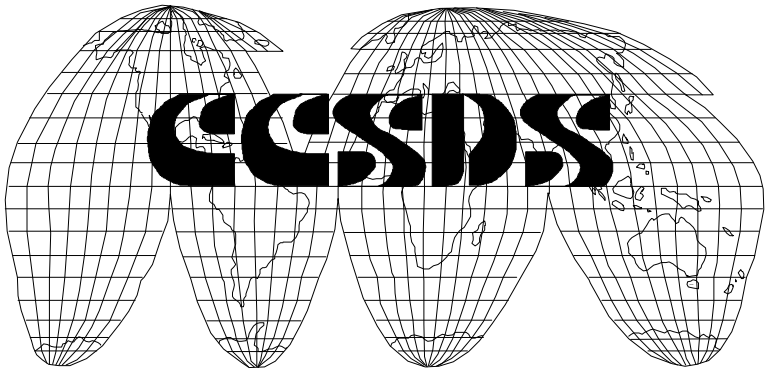
Data Entity Dictionary Specification Language (DEDSL)

(ZCSD0011/ZCSD0012)

CCSDS 6xx.0-W-0.4

WHITE BOOK

December 1995



AUTHORITY

Issue:	White Book, Issue 0.4
Date:	19 December 1995
Location:	TBD

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorisation of CCSDS Recommendations is detailed in reference [14], and the record of Agency participation in the authorisation of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommendation is published and maintained by:

CCSDS Secretariat
Program Integration Division (Code -OI)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organisation officially established by the management of the member space Agencies. The committee meets periodically to address data system problems that are common to all participants and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of the committee are termed **Recommendations** and are not considered binding on any Agency.

This **Recommendation** is issued by, and represents the consensus of, the CCSDS Plenary body. Agency endorsement of the **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accordance with the relevant **Recommendation**. establishing such a **standard** does not preclude other provisions which an Agency may develop.
- Whenever an Agency establishes a CCSDS-related **standard**, the Agency will provide other CCSDS member Agencies with the following information:
  - The **standard** itself.
  - The anticipated date of initial operational capability.
  - The anticipated duration of operational service.
- Specific service arrangements shall be made via memorandum of agreement. Neither this **Recommendation** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommendation** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or cancelled.

In those instances when a new version of a **Recommendation** is issued, existing CCSDS-related Agency **standards** and implementation are not negated or deemed to be non-CCSDS compliant. It is the responsibility of each Agency to determine when such **standards** or implementation are to be modified. Each Agency is, however, strongly encouraged to direct planning of its new **standards** and implementations towards the later versions of the **Recommendation**.

FOREWORD

This Recommendation is a technical Recommendation that provides a model and language to increase the standardisation of the expression of simple semantic concepts that are to be carried with data. These semantic concepts are given standard names and a standard way of expressing them is also provided. The semantic information may be conveyed either in a computer processable manner or via conventional (e.g. paper) documentation.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This document is therefore subject to CCSDS document management and change control procedures which are defined in reference [14].

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page ii.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- National Aeronautics and Space Administration (NASA HQ)/USA.
- National Space Development Agency of Japan (NASDA)/Japan.

Observer Agencies

- Australian Space Office (ASO)/Australia.
- Austrian Space Agency (ASA)/Austria.
- Belgian Science Policy Office (SPO)/Belgium.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organisation for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Industry Canada/Communications Research Center (CRC)/Canada.
- Institute of Space and Astronautical Science (ISAS)/Japan.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status/ Changes
CCSDS 6xx.0-W-0.1	Recommendation Space Data System Standards: Semantic Description Language - Specification	23 June 1995	Issue 0.1
CCSDS 6xx.0-W-0.2	Recommendation Space Data System Standards:-- Data Entity Dictionary Specification Language (DEDSL)	23 August 1995	Issue 0.2
CCSDS 6xx.0-W-0.3	Recommendation Space Data System Standards:-- Data Entity Dictionary Specification Language (DEDSL)	11 October 1995	Issue 0.3
CCSDS 6xx.0-W-0.4	Recommendation Space Data System Standards:-- Data Entity Dictionary Specification Language (DEDSL)	19 December 1995	Issue 0.4

CONTENTS

Sections **Page**

Figures

Page

Tables

Page

1 INTRODUCTION

1.1 Purpose and Scope

The purpose of this document is to establish a specification for a standard technique for describing the meaning of data, in order to interchange and access data in a more uniform and automated fashion.

This Recommendation defines several standard terms (i.e., attribute names), together with constraints on their associated values, that can be used to document the meaning of named data entities, or collections of data entities, found within a data collection. These terms cover simple data semantics such as name, meaning, and units. Since the standard can not address all possible semantic information that can be conceived for all data, conventions are provided in order to allow users to define in a standard manner their project-specific data documenting terms. This Recommendation also defines a standard syntax for expressing the terms and their values.

1.2 Applicability

The specifications in this document are applicable to all space-related science and engineering data exchanges where data descriptions are desired. The use of a formal language to describe data allows the definition of software for automatic interpretation, so that Data Entity Dictionary Specification Language (DEDSL) descriptions and their associated data are more automatically interpretable.

Even though the DEDSL is designed to operate in the frame of the CCSDS developed SFDU concept, its applicability is much wider:

- DEDSL can be used together with any formal syntax data description language (e.g. EAST, see references [1] and [2]);
- DEDSL can be used within any type of conventional interface specification or format specification documents and also within any document format such as ASCII text, word processor format or paper document;
- DEDSL can be used to develop discipline specific dictionaries that provide semantic definitions of all the types of information encountered within a particular domain. This is analogous to a glossary of data types;
- DEDSL can be used in order to standardize attribute features as provided in self-describing data formats, such as Hierarchical Data Format (HDF) and Common Data Format (CDF) and Flexible Image Transport System (FITS).

1.3 Rational

A statement of the problem to be overcome by this Recommendation can be described as follows:

- Many standards deal with the description of the physical layout of space data, i.e. the syntax description. Currently users and projects define customized solutions in order to

document the meaning of the data entities, i.e. the semantic information. This leads to no conformance and hence data generated from different users or projects can not easily be combined in automated processing as their techniques for description are not compatible.

- The readability of the generated data documentation is overly complex as different conventions have been adopted.
- Frequently essential information that is required in order to fully explore and process the data has not been documented, thus leading to a loss of data in the long term. A classical example to highlight the nature of the problem is:
  - Within an interface specification document a particular field is described as being **`processing_time`** without any further details of the semantics of the field provided (assume that the syntax of the field is sufficiently documented). It can easily be conceived that two different teams implementing such a specification, will end up using different time references (e.g. they will use their local time or maybe UTC), one team might interpret the information as providing the start time of the processing, whilst the other interpreters it to mean the end time of the processing.

An important aspect of attempting to unambiguously describe any data to a variety of potential users is the consideration that the data must be easily understandable to the receiver of the data. A recurrent problem with many data is that the users of the data (or the generators) do not unambiguously understand the format and meaning of the data. The data designer must unambiguously described all facets of the data; this must include the physical format and the meaning of each element in the data. Whilst the physical format is often far easier to describe in a formal manner, the meaning of the data elements is frequently incomplete as the designer has assumed a certain level of inherent knowledge in the receiver of the product or has not consider all facets that must be defined.

Furthermore, as a typical user will receive data from many sources and does not wish to write custom software to use and process the data there must be standardisation in the way the products are described so that generic software can be reused as much as possible .

1.4 Document Structure

This document presents the specification in a layered manner. Firstly the abstract definition of the semantic information that is required to be conveyed is defined. This is done so that the actual technique used to convey the information is independent of the information content and therefore the same abstract standard can be used within different formatting environments. This also permits the semantic information to be translated to different representations as may be needed when data are transferred across different domains.

Secondly a standard technique for conveying the information using the CCSDS developed Parameter Value Language (PVL, see references [5] and [6]) is specified. This is the recommended method to use to convey semantic information between a data generator and end user. Finally a number of scenarios are presented that show how this DEDSL specification should be used and how it can be interfaced to both conventional fixed format data and flexible formatting techniques.

In summary, the document is structured as follows:

- Section 2 gives a view of data entities and attributes that should be readily mapable to most data formats, and it provides a context within which to understand the standards specified in Sections 3 and 4.
- Section 3 specifies the abstract semantic description technique, including what information must be conveyed and when it is applicable.
- Section 4 specifies the PVL implementation of the abstract standard; this gives a concrete syntax to the information that is specified in Section 3 and conventions to be used.
- Annex A provides a number of scenarios that describe the usage of this recommendation, what issues can arise and compatibility with existing formatting techniques.
- Annex B provides a list of the terms to be used for units when describing data entities and their representation.
- Annex C defines the ASCII character set that is used in the representation of the DEDSL along with the subset of ASCII that conforms to the Restricted ASCII character set as used within this Recommendation.
- Annex D provides examples of user defined attributes so as to enhance the readers understanding of this concept.

1.5 Definitions

1.5.1 Glossary of Terms

For the purpose of this document the following definitions apply:

Attribute	An attribute is a piece of information that describes another piece of data; this information characterises or enhances understanding of the data that is being described. Attributes are the primary focus of this Recommendation in that they are used to defined the semantics of data.
Data entity	A data entity is the piece of data that is to be described by using attributes and the values of the attributes. A data entity can be of a simple type, such as integer or real; it can also be a compound type consisting of a structure of various other simple and compound entities.
Data entity dictionary	A data entity dictionary is a listing of a number of semantic definitions of various data entities. Each data entity described in the dictionary is described in terms of attributes and attribute values. Data entity dictionaries may be just for a single product, i.e. all the data entities within a single product are described in a corresponding single dictionary, or the data entity dictionary may be a discipline oriented dictionary that holds a number of previously defined data entity definitions which may be used by data definers and users for reference.
Data type	In the context of this document, data type is an indication of the conceptual scientific form of the data entity, such as integer, real, string,

	sequence of entities, etc.
DEDSL module	The DEDSL module is the combination of the DEDSL identification information, which provides information about the complete DEDSL module, the definition of any user defined attributes and the definition of the actual data entities. Various parts of a DEDSL module are optional. In practical terms the DEDSL module could be either a file, SFDU LVO value field (see references [3] and [4]), etc.
Dewy number	A Dewy number is a number which contains a number of decimal points so as to indicate levels of significance. For example, in the Dewy number '2.1.3', '2' is the highest level of significance down to '3' being the lowest level of significance. Dewy numbers are frequently used to indicate versions of documents or software.
Module identification attribute	An attribute that carries information that identifies the complete DEDSL module. This information does not describe the data entities in any way, but purely the complete file or data block.
Quoted string	A quoted string is a string of characters that are enclosed within quotation marks. These may be either single or double quotation marks (although the same at each end of the string). White space characters are permitted within the quoted string (see references [5] and [6] for an exact definition).
Semantics	Semantics are the information that describes the meaning of data rather than the physical representation of that data. Semantics potentially cover a very large domain, from the simple such as the units of a data entity to the complex, such as the relationship of a data entity to another.
Standard attribute	A standard attribute is one that is defined here within the Recommendation. When the DEDSL is used there are standard attributes that are mandatory and others which are optional, but the same attribute name cannot be redefined if the semantic information is to be in conformance with this Recommendation.
Syntax	Syntax is the definition of the physical representation of data. It includes the structural arrangement of the fields within data, the physical hardware representation and results in a clear understanding of the abstract values of the data.
Unquoted string	An unquoted string is a string of characters that do not contain any white space characters and hence is delimited on either side by any white space characters, e.g. **`NAME,`** (see references [5] and [6] for an exact definition).
User defined attribute	A user defined attribute is an attribute that is defined by a particular user or project and after definition is then used in the same manner as a 'standard attribute'. This Recommendation specifies a standard method of defining user defined attributes so as to produce unambiguous definitions and encourage reuse of user defined attributes.
White space	White space is defined to consist of the ASCII characters line feed ($0A_{hex}$), carriage return ($0D_{hex}$), horizontal tab (09_{hex}), vertical tab ($0B_{hex}$),

form feed ($0C_{hex}$) and space (20_{hex}).

1.5.2 Nomenclature

The following conventions apply throughout this Recommendation:

a) The words “shall” and “must” imply binding and verifiable specification;

b) The word “should” implies an optional, but desirable, specification;

c) The word “may” implies and optional specification;

d) The words “is”, “are” and “will” imply statements of fact.

1.5.3 Conventions

This document uses structure diagrams to illustrate the structure of the DEDSL constructs. Components of a construct are called elements. The following conventions are used:

- The element named on the left of the ::= symbol is the element being defined.
- The diagram on the right of the ::= symbol is the corresponding definition.
- Elements that are presented in bold characters in a rounded box are PVL statements that are defined elsewhere in the recommendation.
- Elements that are presented in italic characters in a square box are elements that are further defined in another structure diagram.
- A vertical branch represents a choice.
- A repetition is indicated by a loop-back covering the object to be repeated.

The following example presents a diagram specifying the declaration of **‘Element’**. **‘Element’** is defined as first **‘Another element’**; this element can be repeated any number of times due to the loop back over the element (i.e. one or more ‘ **Another element**’s must be included). **‘Another element’** would be defined in another structure diagram. Following **‘Another element’**, there is a choice between **‘STATEMENT_1=’**, **‘STATEMENT_2=’** or nothing. **‘STATEMENT_1=’** and **‘STATEMENT_2=’** both represent PVL statements specified elsewhere in the document. Following this there is **‘STATEMENT_3=’**, which represents another PVL statement specified elsewhere in the document, and finally ‘ **STATEMENT_4=’** is another PVL statement that can either be passed by altogether or included any number of times by virtue of the loop-back. (i.e. zero or more ‘ **STATEMENT_4=**’s).

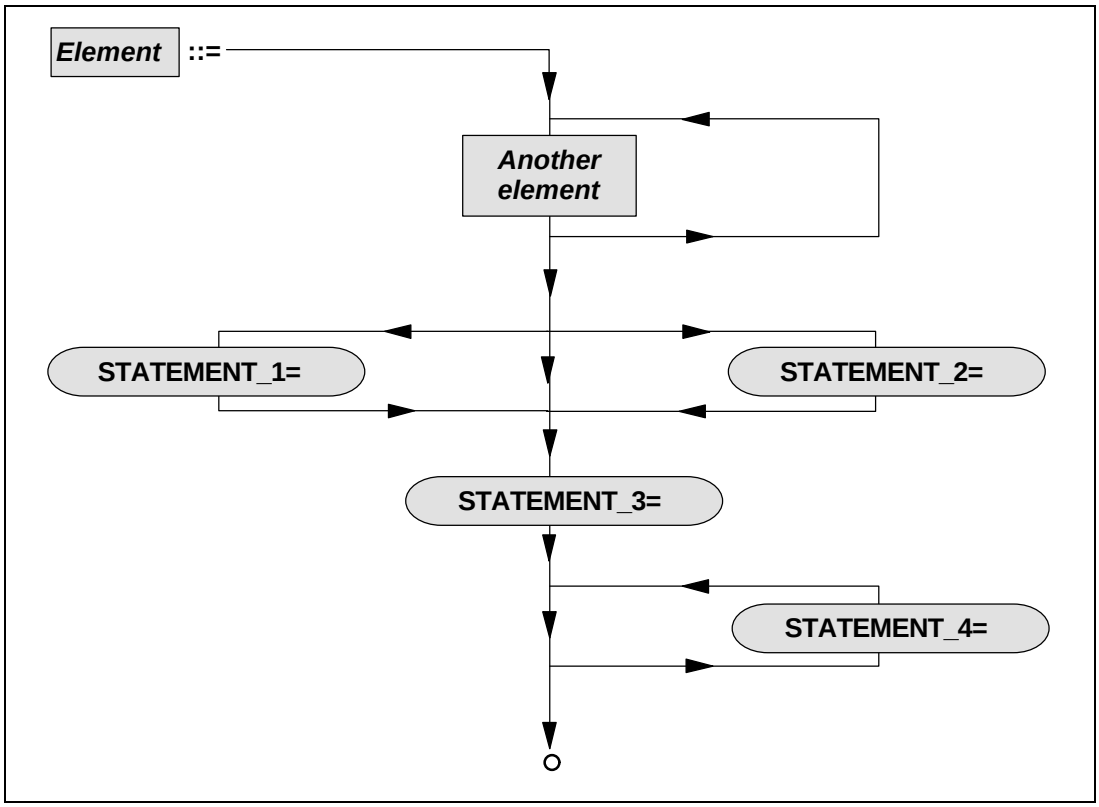


Figure 1-1: Example Structure Diagram

This document uses examples to illustrate the DEDSL. The following conventions are used in the examples:

- All examples of DEDSL are shown in a bold, fixed spacing font;
- Any parts of the syntax that are variable and defined when used are shown in lower-case within angle brackets, e.g. **`<quoted string>`**.

1.6 References

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently available CCSDS Recommendations.

[1] ***The Data Description Language EAST -- Specification (CCSD0010)***, Recommendation for Space Data System Standards, CCSDS 644.0-W-0.4, White Book, Issue 0.4, CCSDS, March 1995.

[2] ***The Data Description Language EAST -- A Tutorial***, Report Concerning Space Data System Standards, CCSDS 644.0-W-0.4, White Book, Issue 0.4, CCSDS, March 1995.

[3] ***Standard Formatted Data Units - Structure and Construction Rules***, Recommendation for Space Data System Standards, CCSDS 620.0-B-2, Blue Book, Issue 2, CCSDS, May 1992.

[4] ***Standard Formatted Data Units - A Tutorial***, Report Concerning Space Data System Standards, CCSDS 621.0-G-1, Green Book, Issue 1, CCSDS, May 1992.

[5] ***Parameter Value Language Specification (CCSD0006),*** Recommendation for Space Data System Standards, CCSDS 641.0-B-1, Blue Book, Issue 1, CCSDS, May 1992.

[6] ***Parameter Value Language - A Tutorial,*** Report Concerning Space Data System Standards, CCSDS 641.0-G-1, Green Book, Issue 1, CCSDS, May 1992.

[7] ***ASCII Encoded English (CCSD0002)***, Recommendation for Space Data System Standards, CCSDS 643.0-B-1, Blue Book, Issue 1, CCSDS, November 1992.

[8] ***Information Processing - Representation of numerical values in character strings for information interchange***, ISO 6093-1985.

[9] ***Standard Formatted data Units -- Control Authority Procedures***, Recommendation for Space Data System Standards, CCSDS 630.0-B-1, Blue Book, Issue 1, CCSDS, June 1993.

[10] ***Standard Formatted data Units -- Control Authority Procedures Tutorial***, Report Concerning Space Data System Standards, CCSDS 631.0-G-2, Green Book, Issue 2, CCSDS, November 1994.

[11] ***UNIDATA Units Package***, NCAR, Version 1.10.2, 19 October 1995, (http://www.unidata.ucar.edu/packages/udunits/index.html).

[12] ***Information Technology - Open systems Interconnection - Specification of Abstract Syntax Notation.One (ASN.1),*** ISO/IEC 8824, 2nd issue, 15 Dec 1990.

[13] ***Time Code Formats,*** Recommendation for Space Data System Standards, CCSDS 301.0-B-2, Blue Book, Issue 2, CCSDS, April 1990.

[14] ***Procedures manual for the Consultative Committee for Space Data Systems***, CCSDS A00.0-Y-6, Yellow Book, Issue 6, CCSDS, May 1994.

2 OVERVIEW

Sections 3 and 4 defines standards to be used for the documentation of semantic information (or attributes) of data entities contained in formatted data. This section is intended to provide a sufficient (but limited) view of data entities and their attributes so that users of this document can readily understand how the simple semantics of Sections 3 and 4 may be used with their own data to improve wider understanding. As an additional aid, Annex A provides some example applications in the form of scenarios.

A data entity can be defined as a having three basic components:

- A name which uniquely identifies a data entity and maps to a specific location within some data;
- A value which is present at the mapped location within the data;
- A set of zero or more attributes that provide information about the entity or its value.

As applied to a data structure, a data entity may have multiple instances within the product. A data entity is also a recursive concept, that is, a data entity may be composed of other data entities. The value of a data entity may be a single element of formatted data such as a basic type (e.g. integer) or a collection of aggregated data elements (e.g. structure, array, record, file, etc.). The key idea behind the identification of a data entity is that users require access to the value(s) of a data entity by use of a data entity identifier (i.e. a name). In many data structures, some of contained data will be understood to provide information about the characteristics or aspects of a data entity (i.e. attributes), rather than the data instance value itself.

For example, the data in Figure 2-1 is a data entity that consists of a sequence of data entity values that provide magnetic field values and their observation times.

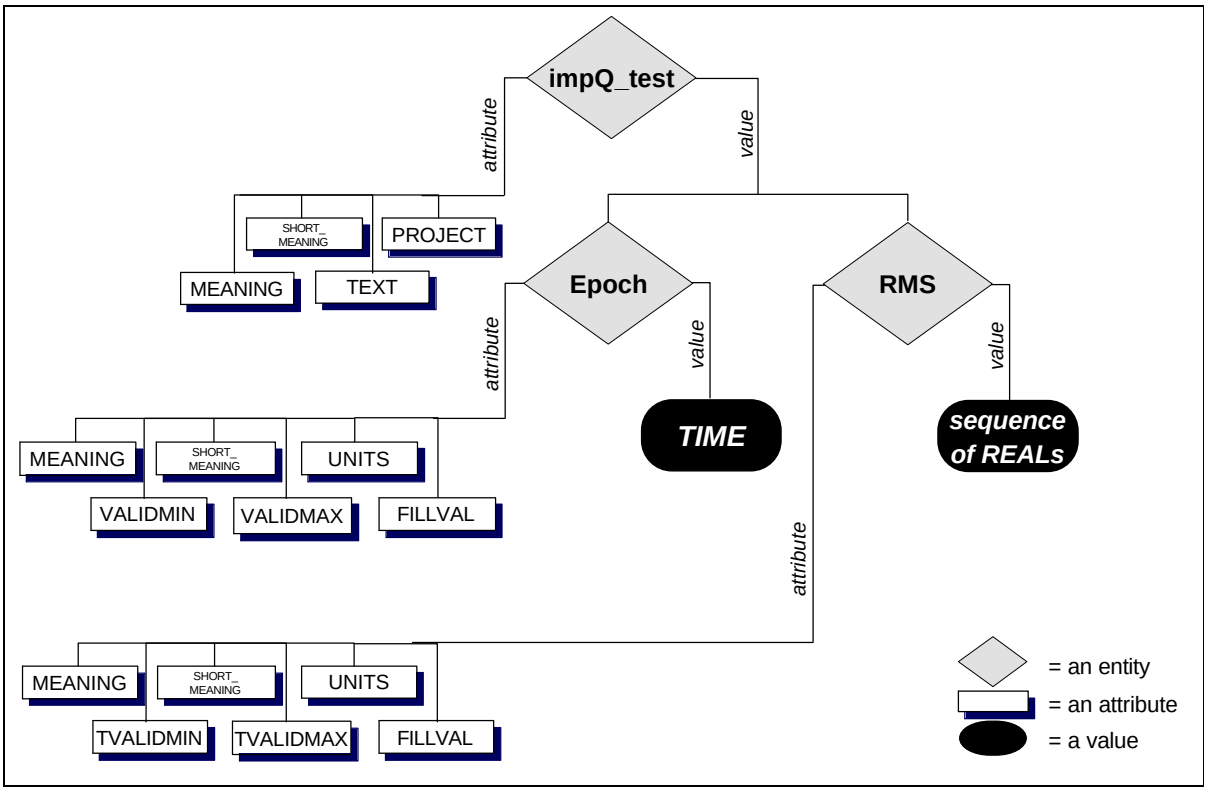


Figure 2-1: Example Hierarchical Data Structure

This entity has an identifier of **impQ_test**. Its value is a sequence of two entities; **Epoch**, which gives the time of the observation and **RMS**, which gives the strength of the magnetic field as a sequence of the components along the x, y, and z co-ordinates. Each entity is further defined by a set of attributes which may include a textual description of the meaning of the values (attribute identifier = **MEANING**), a suggested entity title of short description (attribute identifier = **SHORT_MEANING**), an indication of the units in which the values are expressed (attribute identifier = **UNITS**), and other attributes giving more detailed description of the values and their context. Figure 2-1 shows the hierarchical view of this example which is more fully described in Annex A.2. This example will be used throughout Sections 3 and 4 to indicate how the DEDSL can be used to specify and standardise attribute definitions. Note that some of the attributes (i.e. **MEANING** and **SHORT_MEANING**) are used in the description of all entities, whilst others, such as **UNITS,** are used only to describe entities that have a scalar type as a value.

The DEDSL module that defines the simple semantics for the example entity shown in Figure 2-1 would look like the following. It is not anticipated that the reader shall be able to understand this example completely at this stage, but it serves as a guide to what the following sections of the document shall specify. All examples in the following sections shall be taken from the complete entity example:

```
/*************************************************************
**  DEDSL module identification information                **
*************************************************************/

BEGIN_GROUP = MODULE_IDENTIFICATION ;
```

```
    DEDSL_VERSION = 0.1 ;
    MODULE_TITLE = “impQ_test dictionary” ;
    MODULE_ADID = NSSD1393 ;
END_GROUP = MODULE_IDENTIFICATION ;

/*************************************************************
**  Definition of user defined attributes                   **
*************************************************************/

BEGIN_GROUP = ATTRIBUTE_DEFINITION ;
    ATTRIBUTE_NAME = PROJECT ;
    ATTRIBUTE_MEANING = “This attribute specifies the name of the project
                         for which the data entity attribute has been
                         defined.” ;
    ATTRIBUTE_VALUE_SYNTAX = STRING ;
    ATTRIBUTE_OCCURANCE = 0..1 ;
    ATTRIBUTE_COMMENT = "A formal list of recognised project names can be
                         found out by contacting J. Smith, within
                         NASA/NSSDC” ;
END_GROUP = ATTRIBUTE_DEFINITION ;

BEGIN_GROUP = ATTRIBUTE_DEFINITION ;
    ATTRIBUTE_NAME = VALIDMAX ;
    ATTRIBUTE_MEANING = “This is a single scalar value that indicates
                         the maximum value that the entity may be
                         within any single instance of the data
                         structure.” ;
    ATTRIBUTE_VALUE_SYNTAX = TIME ;
    ATTRIBUTE_OCCURANCE = 0..1 ;
END_GROUP = ATTRIBUTE_DEFINITION ;

       /* Similar user defined attribute definitions for   **
       ** VALIDMIN, TEXT, FILLVAL, TVALIDMIN and TVALIDMAX */

/*************************************************************
**  Entity definitions in terms of standard attributes and    **
**  previously stated user defined attributes                 **
*************************************************************/

BEGIN_GROUP = ENTITY_DEFINITION;
    NAME = impQ_test ;
    MEANING= "This entity contains magnetic field observations from
              Interplanetary Monitoring Platform Q (IMP-Q) as
              prepared for testing.  There will probably be multiple
              instances of this entity having the name ‘impQ_test’";
    SHORT_MEANING = “IMP-9 Magnetic Field" ;
    PROJECT = "International Solar-Terrestrial Physics" ;
    TEXT = "Data: 1.024 minute averages impQ test values” ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
    NAME = Epoch ;
    MEANING = "Time of mid-points of the observation intervals since
               A.D." ;
    SHORT_MEANING = “Time Line" ;
    UNITS = “ms” ;
    ALIAS = (TIME_LINE, “This entity is identified by this name within
```

```
                          the ESA ground segment, rather than the primary
                          name within the NASA ground segment" );
    SPECIFIC_INSTANCE = ( 19920812T15:54:23.12, "Experiment switch on
                           time" );
    COMMENT = "This is an example entity to demonstrates the DEDSL" ;
    VALIDMIN = 19900101T00:00:00.00 ;
    VALIDMAX = 20101231T23:59:59.00 ;
    FILLVAL = -10.0e30 ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION ;
    NAME = RMS ;
    MEANING = "Components of B in GSE coordinates obtained by taking
               the root-mean-square of the values in the observing
               intervals" ;
    SHORT_MEANING = "Components of RMS of B (GSE)" ;
    UNITS = "n tesla" ;
    TVALIDMIN = (0.0, 0.0, 0.0) ;
    TVALIDMAX = (20.0, 20.0, 20.0) ;
    FILLVAL = -10.0E30 ;
END_GROUP = ENTITY_DEFINITION;

/*************************************************************
/**                End of data entity dictionary               **
*************************************************************/
```

Some of the critical issues in the description of data entities are associated with the assignment of names. There are two conflicting requirements that affect the assignment of names to data entities. The first is to allow a name to translate into a unique location within the data to obtain the correct value, while the second is to use the same name when referring to the same real world concept. This leads to the concept of hierarchical naming where the complete name of a data entity includes the names of all the entities it is contained within. For example in Figure 2-1 the entity **Epoch** is contained within the entity **impQ_test**. The full name for this entity is **impQ_test.Epoch** (Note the period is a convention adopted within this recommendation to indicate the hierarchical relationship of 'contained within').

The techniques discussed in Sections 3 and 4 can be applied to any named data item so a project or data designer can associate attributes with **Epoch** that should apply to any entity whose complete name contains the named element **Epoch** at its lowest hierarchical level (i.e. after the last period). This capability is in addition to the more common usage of this recommendation which is the description of data entities which have unique complete names.

An idealised access to data can be viewed as an object request as shown in Figure 2-2 .

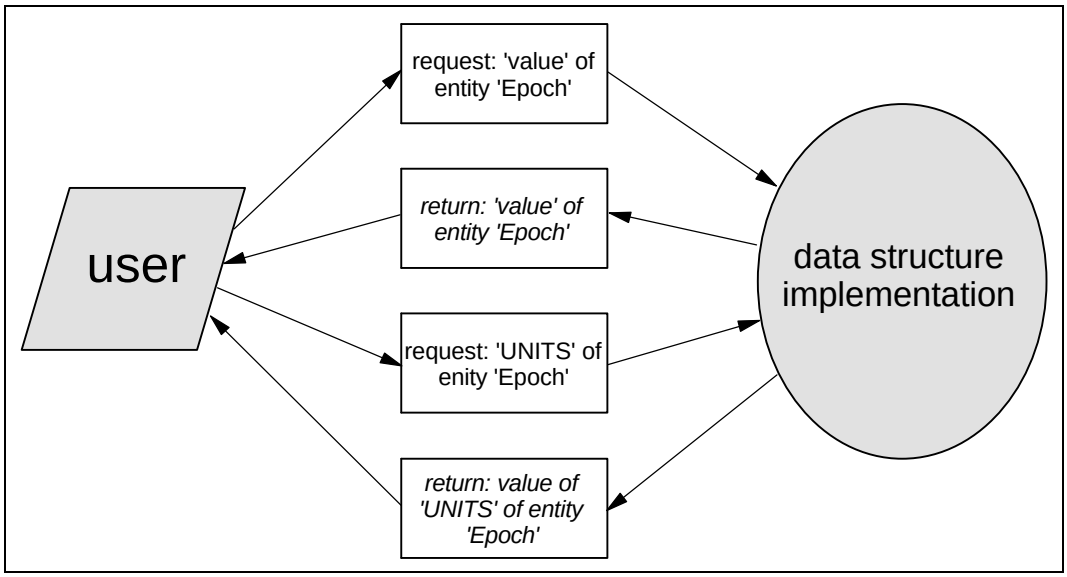


Figure 2-2: Object Request View of Data

The data consumer sends a message to the data interface requesting any or all of the attributes of a data entity with a particular identifier. The methods interfacing to the data assemble the requested names and values and return them to the data consumer. The methods for gathering the attribute values are dependent upon the data product storage techniques implemented.

There are two problems that prevent the idealised interface presented in Figure 2-2 from being practical in the present day.

- Firstly, many existing products do not 'contain' machine interpretable attributes. If the attribute information does exist, they are frequently in paper documentation or notes which are not computer accessible.
- Secondly, when the attributes do exist in computer accessible form, there are inadequate standard identifiers for the attributes so that most consumer software does not know what messages to send to the interface.

This recommendation (in Section 3) attempts to standardise the names for some of the attribute identifiers. If data producers use these standard attribute names it will enable the creation of standard consumer software to query data with standard messages. Section 3 also defines a technique for a user to define their own attributes in a formal manner and hence encourage reuse and greater interoperability.

This recommendation (in Section 4) defines a concrete syntax to allow the expression of attributes (and user defined attributes) in a standard manner and to facilitate their addition to data. One approach to linking these attributes with data is through the use of the SFDU packaging technique (see references [3] and [4]) with a DEDSL module. This is discussed briefly in the scenario presented in Annex A.1: Using the DEDSL with a Formal Data Description Language '.

3 ABSTRACT STANDARD

The semantic information required to describe a data entity is seen as a collection of attributes. Each attribute describes a particular semantic characteristic of the data entity.

In the following sub-sections, the collection of attributes available to describe the basic semantics of a data entity are identified. These attributes fall into two categories; 'standard attributes' that are defined by this recommendation and 'user defined attributes' that are defined by the user in a standardised way, but which can then be used in the same manner as 'standard attributes'.

In order to facilitate the readability of attributes and support cross-comparison of data, each attribute requires a standard identifier as well as standard conventions for the specification of the attribute's value.

3.1 Standard Attributes

3.1.1 Data Entity Identification: NAME

Purpose This attribute may be used to identify the data entity that is being described. The value of this attribute is an identifier that may be used to link a collection of attributes with an equivalent identifier in, or associated with, the data entity.

The value of this attribute may also be used by the software developer to name corresponding variables in software code or as a field to be searched on for location of particular data entities.

Term to be used The standard term to be used for this attribute is: **NAME**

Occurrence This attribute is mandatory and can only appear once for each data entity described.

Convention for the attribute value For the value of this attribute the following conventions must be followed:

- it shall be a free format character string;
- no white space characters [1] are permitted in the value;
- the value must to be unique within the DEDSL module (see Section 4.1);
- the value is case sensitive. *Note, whilst this means that two values of NAME may only differ by case of certain letters, this practice is discouraged.*
- when identifying entities within a hierarchical data structure, the

[1] *For the purpose of this recommendation, white space characters are defined as the following characters from the ASCII character set: line feed ($0A_{hex}$), carriage return ($0D_{hex}$), horizontal tab (09_{hex}), vertical tab ($0B_{hex}$), form feed ($0C_{hex}$) and space (20_{hex})(see reference [5]).*

commonly used 'dot' notation should be used whenever possible. *For example,* ***impQ_test.Epoch*** *or* ***impQ_test.RMS.***

- the maximum length of the value of this attribute is 400 characters.

Example attribute value **Epoch**

3.1.2 Full Textual Description of a Data Entity: MEANING

Purpose Required to give the description and meaning of the data entities identified by the **NAME** attribute. This attribute is intended for human readership and therefore any information that will increase the understanding of the identified data entity should be included.

It is intended that the value of this attribute can be of significant length and hence provide as complete a description of the data entity as possible. The value of this attribute can be used as a field to be searched on for the location of particular data entities.

The value of this attribute may include, in natural language, the same semantic information that is carried in a more formal manner by other attributes. This is neither a requirement or illegal, but the user must make sure inconsistencies do not arise.

Term to be used The standard term to be used for this attribute is: **MEANING**

Occurrence This attribute is mandatory and can only appear once for each data entity described.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- it shall be a free format character string that can span a number of lines;
- it may include any number of white space characters;
- the maximum length of the value of this attribute is 8000 characters.

Example attribute value **Time of mid-points of the observation intervals since A.D.**

3.1.3 Short Textual Description of a Data Entity: SHORT_MEANING

Purpose The value of this attribute provides a short concise description and meaning of the data entity identified by the **NAME** attribute. This attribute provides a summary of the more detailed information provided by the **MEANING** attribute

The value of this attribute can be used as a field to be searched on for the location of particular data entities. It is also intended to be

used for display purposes by automated software, where the complete **MEANING** value would be too long to present in a convenient manner to the user.

Term to be used The standard term to be used for this attribute is: **SHORT_MEANING**

Occurrence This attribute is optional, but if provided it can only appear once for each data entity described.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- it shall be a free format character string of no more than one line;
- it may include any number of space characters (ASCII 20_{hex});
- the maximum length of the value of this attribute is 80 characters.

Example attribute value **Time line**

3.1.4 Scientific Units of a Data Entity: UNITS

Purpose The value of this attribute specifies the scientific units that should be associated with the value of the data entity so as to make the value meaningful in the 'real-world'.

Term to be used The standard term to be used for this attribute is: **UNITS**

Occurrence If the data entity that is being defined is of a scientific scalar type then this attribute is mandatory and may appear only once. If the data entity is non-scalar then the attribute shall not be specified.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- it shall be a character string of no more than one line;
- it may include any number of space characters (ASCII 20_{hex});
- the maximum length of the value of this attribute is 80 characters.

The contents of the character string must conform to the units standard as defined by the NCAR software package Unidata (see reference [11] for more details), that being:

"A unit is specified as an arbitrary product of constants and unit names raised to arbitrary integral powers. Division is indicated by a slash '/'. Multiplication is indicated by white space (see footnote 1, on page 13), a period '.', or a hyphen '-'. Exponentiation is indicated by an integer suffix or by the exponentiation operators '^' and '**'. Parentheses may be used for grouping and disambiguation.

Arbitrary Galilean transformations (i.e. y = ax + b) are supported. In particular, temperature and time conversions are correctly handled.

The specification:

degF @ 32

indicates a Fahrenheit scale with the origin shifted to thirty-two degrees Fahrenheit (i.e. to zero degrees Celsius). The Celsius scale is equivalent to the following unit:

(1.8 degF) @ 32

Note that the origin-shift operation takes precedence over multiplication. In order of increasing precedence, the operations are division, multiplication, origin-shift, and exponentiation."

The aim of standardising on the NCAR Unidata format for units is so that automated software can be used to process and convert units. This permits users to readily process data from different sources when supplied with different units.

The standard set of unit names and prefixes that are permitted are shown in Annex B. Only the indicated representations can be used within the value of this attribute, even though the Unidata software can support a wider range of unit names. The configuration file for the NCAR Unidata units software which enforces the conventions specified in the section above can also be found in Annex B.

Example attribute values **ms**

3.1.5 Alternate Identification of a Data Entity: ALIAS

Purpose The value of this attribute provides an alternative identifier of the data entity that may be required for the purpose of compatibility with historical data or data deriving from different sources. For example, different sources may produce data which includes the same entities, but which have different names. Through the use of this attribute it will be possible to define the semantic information only once. Along with the alternate identifier, this attribute value shall provide a description of the context of when the alternate identifier is used.

The value of the alternate identifier can be searched when an identifier used in a corresponding syntax description is not found within the **NAME** values.

Term to be used The standard term to be used for this attribute is: **ALIAS**

Occurrence This attribute is optional, although if it does appear it may appear any number of times. Each alternate identifier must be unique within the scope of the DEDSL module.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- there shall be two values associated with this attribute: an alternative identifier and a context description;

- the alternate identifier must follow the same conventions as for that of the **NAME** attribute ;
- the context information shall be a free format character string that can span a number of lines and contain any number of white space characters;
- that the maximum length of the context information string is 400 characters.

Example attribute value

Alternate identifier: **TIME_LINE**

Context information: **This entity is identified by this name within the ESA ground segment, rather than the primary name within the NASA ground segment.**

3.1.6 Specific Meaning of an Instance of a Data Entity: SPECIFIC_INSTANCE

Purpose

The value of this attribute provides a real world meaning for a specific instance of the value of the data entity being described. The reason for providing this information is so that the user can see that there is some specific meaning associated with a particular value instance that indicates something more than just the abstract value. For example, the fact that zero degree latitude is the equator could be defined. This means that the value of this attribute must provide both an instance of the entity value and a definition of its specific meaning.

The values of the attribute can be used to enhance user interfaces and therefore user understanding. For example, instead of displaying to the user the abstract value of an entity, the 'system' could first check the DEDSL definition to see if there is a specific meaning for this abstract value, and if there is, display the specific meaning string instead. Likewise, a user could enter a meaning definition by name, e.g. **equator**, and the 'system' automatically (via the DEDSL definition) translate this name to a specific instance value.

Term to be used

The standard term to be used for this attribute is:
SPECIFIC_INSTANCE

Occurrence

This attribute is optional, although if is does appear it may appear any number of times. If the attribute does appear more than once for any single entity, then each instance value that is specified must be unique as must each specific meaning definition.

Convention for the attribute value

For this attribute the following conventions must be followed for its value:

- there shall be two values associated with this attribute: an instance value and a specific meaning definition.;
- the specific instance value must be expressed in ASCII (i.e. for a binary number, the ASCII representation is as defined in

reference [8]) and be no longer than 80 characters;

- the specific meaning definition:
  - shall be a free format character string of no more than one line;
  - it may include any number of space characters (ASCII 20_{hex});
  - the maximum length of the specific meaning string is 80 characters

Example attribute value

Specific instance: **`19920812T17:54:23.12`**

Context information: **`Experiment switch on time`**

3.1.7 Associated Information about a Data Entity: COMMENT

Purpose

This attribute is for providing information which is not directly required to understand the real world meaning of the data entity, but which could still assist the user of the data in some manner. This attribute is intended for human readership.

Term to be used

The standard term to be used for this attribute is: **`COMMENT`**

Occurrence

This attribute is optional and can appear any number of times. This is so that additional information can be added at any time to the entity definition without having to edit existing attributes.

Convention for the attribute value

For this attribute the following conventions must be followed for its value:

- it shall be a free format character string that can span a number of lines;
- it may include any number of white space characters;
- the maximum length of the value of this attribute is 8000 characters.

Example attribute value

`This entity definition is based upon that defined by the Planetary Data System developed by NASA/JPL. For further details about the PDS please contact JPL directly.`

3.2 User Defined Attributes

The standard attributes specified in Section 3.1 are those predefined by this recommendation and must be recognised by any system that states conformance to this recommendation. It is recognised that there may be further attributes that are more specific to a particular domain, mission or project. This section defines a means where by users may define their own specific attributes and then use these attributes in the same manner as the standard attributes to define the semantics of their particular data entities. These attributes shall be called ‘user defined attributes’.

Just as it is important to unambiguously define data entity semantics it is also important to define unambiguously the definition of user defined attributes. To do this a similar technique shall be used as that used to define the entity semantics shown in Section 3.1, where a number of parameters are defined which are required to characterise the user define attributes.

3.2.1 Identification of an Attribute: ATTRIBUTE_NAME

Purpose Specifies the user defined attribute identifier that shall be used. This identifier must be unique within any single DEDSL module, that is, it must not be the same as a standard attribute identifier and cannot be a repetition of a already existing user defined attribute identifier.

Term to be used The standard term to be used for this parameter is:
`ATTRIBUTE_NAME`

Occurrence This parameter is mandatory and can only appear once for each user define attribute definition.

Convention for the parameter value For this parameter the following conventions must be followed for its value:

- no spaces are permitted in the value;
- the value must to be unique within the scope of the DEDSL module;
- the value is case sensitive. *Note, whilst this means that two values of* ***`ATTRIBUTE_NAME`*** *may only differ in case of certain letters, this practice is discouraged.*
- the maximum length of the value of this parameter is 40 characters.

Example parameter value **`PROJECT`**

3.2.2 Full Textual Description of an Attribute: ATTRIBUTE_MEANING

Purpose Required to give the description and meaning of the user define attribute identified by the **`ATTRIBUTE_NAME`** parameter. This parameter is intended for human readership and therefore any information that will increase the understanding of the new user defined attribute should be included.

It is intended that the value of this parameter can be of a significant length and hence provide as complete a description of the user defined attribute as is possible.

Term to be used The standard term to be used for this parameter is:
`ATTRIBUTE_MEANING`

Occurrence This parameter is mandatory and can only appear once for each user defined attribute definition.

Convention for the parameter value For this parameter the following conventions must be followed for its value:

- it shall be a free format character string that can span a number of lines;
- it may include any number of white space characters;
- the maximum length of the value of this parameter is 8000 characters.

Example parameter value **`This attribute provides the name of the project which has defined the data entity.`**

3.2.3 Syntax of an Attribute Value: ATTRIBUTE_VALUE_SYNTAX

Purpose This parameter specifies the syntax of the value of the user defined attribute.

Term to be used The standard term to be used for this parameter is:
`ATTRIBUTE_VALUE_SYNTAX`

Occurrence This parameter is mandatory and can only appear once for each user define attribute definition.

Convention for the parameter value This parameter value can only be one of the following strings which corresponds to an attribute value as indicated:

`INTEGER`	a mathematical integer number formatted according to ISO NR1 format (see reference [8])
`REAL`	a mathematical real number formatted according to ISO NR2 or NR3 format (see reference [8])
`NUMERIC`	a mathematical number formatted according to ISO NR1, NR2 or NR3 format (see reference [8])
`STRING`	a quoted character string character string that may contain any character from the printable ASCII character set (see Annex C)
`LITERAL`	an unquoted character string that can contain any characters from the printable ASCII character set except for white space characters (see Annex C)
`TIME`	a date/time value formatted according to the ISO/CCSDS standard for time/date formats (see reference [13])
`ENUMERATION`	a list of discrete values that the attribute can take. The possible values must be listed in the value of the **`ATTRIBUTE_MEANING`** parameter.
`SEQUENCE`	a sequence of values that can each be of any of types indicated in the above list. There is no limitation on the number of values in the

sequence or the combination of values types. The **ATTRIBUTE_MEANING** value must clearly explain the meaning of each value in the sequence and the order (whether significant or not)

It is feasible that a more complex description is required , if this is so then a parameter value of **EXTERNAL** should be given. This indicates that the description of the attribute value is registered externally as specified by the **ATTRIBUTE_EXTERNAL** statement (see below).

Example parameter value **STRING**

3.2.4 External Syntax of an Attribute Value: ATTRIBUTE_EXTERNAL

Purpose This parameter is used to indicate a unique identifier, of the description of the value of the user defined attribute, that is registered externally. The identifier shall be an ADID registered at a Control Authority (see References [9] and [10]).

Term to be used The standard term to be used for this parameter is: **ATTRIBUTE_EXTERNAL**

Occurrence If the value of the **ATTRIBUTE_VALUE_SYNTAX** is **EXTERNAL**, then this attribute is mandatory and can appear only once for each user define attribute definition. If the value of the **ATTRIBUTE_VALUE_SYNTAX** is otherwise, then this attribute shall not appear.

Convention for the parameter value The format of this parameter value is an unquoted ASCII string of eight consecutive Restricted ASCII [2] characters that constitute a registered ADID (see references [9] and [10]).

Example parameter value **EESA1234**

3.2.5 Occurrences of an Attribute: ATTRIBUTE_OCCURANCE

Purpose This parameter specifies the number of times that the user defined attribute may occur when used within the definition of a single data entity. For example the standard attribute **UNITS** can only appear once for any single data entity definition, whilst **ALIAS** is optional, but if it does appear it can then appear any number of times. Similarly, if the user defined a new user defined attribute called **DEFINER**, to be used to indicate the person that defined the entity definition, this could be indicated as mandatory and appearing only once.

Term to be used The standard term to be used for this parameter is:

---

[2] *For a definition of the characters that constitue the Restricted ASCII (RA) character set see Annex C.*

ATTRIBUTE_OCCURANCE

Occurrence This parameter is mandatory and can only appear once for each user define attribute definition.

Convention for the parameter value The format of the parameter value is a string that is defined by the following BNF definition:

a..[b|"n"]

Where **a** is the minimum number of times that the user defined attribute may occur, **b** is the maximum number of times that the user defined attribute may occur or the ASCII character " **n**" indicates that there is no upper limit on the number of times that the user defined attribute may occur. There are no spaces permitted and **a** must be less than or equal to **b**.

For example: **1..1** exactly once

0..1 optional

0..n optional but no specified maximum

1..n at least once but no specified maximum

0..5 optiona l with a maximum of 5 times

1..5 at least once with a maximum of 5 times

3..8 at least 3 times with a maximum of 8 times

It is recommended that a realistic maximum number of occurrences is always specified so that practical software can be designed to handle the user defined attribute.

Example parameter value **0..1**

3.2.6 Example of a Value of an Attribute: ATTRIBUTE_VALUE_EXAMPLE

Purpose This parameter provides an example of a value of the user defined attribute as it would appear in an entity definition.

Term to be used The standard term to be used for this parameter is: **ATTRIBUTE_VALUE_EXAMPLE**

Occurrence This parameter is optional and may appear any number of times demonstrating different examples in each case.

Convention for the parameter value The format of this parameter value must conform with the value syntax as defined by the **ATTRIBUTE_VALUE_SYNTAX** parameter. The maximum length of this parameter value is also dictated by the **ATTRIBUTE_VALUE_SYNTAX** specification.

Example parameter value **ISTP**

3.2.7 Associated Information about an Attribute: ATTRIBUTE_COMMENT

Purpose This parameter is for providing information which is not directly required to understand the meaning of the user define attribute, but which could still assist the user of the user defined attribute in some manner.

Term to be used The standard term to be used for this parameter is:
`ATTRIBUTE_COMMENT`

Occurrence This parameter is optional and can appear any number of times. This is so that additional information can be added at any time to the user defined attribute definition without editing existing parameters..

Convention for the parameter value For this parameter the following conventions must be followed for its value:

- it shall be a free format character string that can span a number of lines;
- it may include any number of white space characters;
- the maximum length of the value of this parameter is 8000 characters.

Example attribute value **`The formal list of recognised project names can be found out by contacting J. Smith, within NASA/NSSDC.`**

3.2.8 Registration of User Defined Attributes

So as to aim for maximum reuse and hence interoperability across missions, projects and agencies, it is desirable that any new user defined attributes that are created by projects are submitted for central registration. This means that they can be reused by other projects, eventually leading to a uniform data entity dictionary across many mission and projects.

The advantage of commonality of the data entity dictionary is that software can be developed to handle the entity definitions, which can then be reused by many other projects.

To register user defined attributes, the data description registration capabilities detailed in the CCSDS Recommendations on Control Authorities (se references [9] and [10]) should be followed.

When a user registers a user defined attribute, the following information must be included:

- Identification of the user - this information shall be as defined in the registration of data descriptions with the CCSDS Control Authority (see references [9] and [10]).
- A specification of the user defined attribute following the conventions defined in Sections 3.2.1 to 3.2.7 .
- If software is available to support processing of the value of the user defined attribute, this should be submitted with the definition.

3.3 DEDSL Module Identification Attributes

It is frequently useful to be able to identify information about a particular DEDSL module, whether the module defines a data entity dictionary intended for a single data structure or whether it is a discipline oriented dictionary. So as to provide this type of information in a coherent and common manner this section identifies a number of attributes that can be considered global to the complete product or domain. These attributes are called Module Identification Attributes.

The primary purpose in providing this information is so that automated software can use it to detect whether it can process the particular version of the DEDSL module.

To convey the necessary information the following attributes are required. They are all ASCII strings.

3.3.1 DEDSL Version Information: DEDSL_VERSION

Purpose This parameter indicates the version of the DEDSL Recommendation that the DEDSL module conforms to. All versions are backwards compatible with previous versions. The version is indicated on the front cover of the Recommendation.

Term to be used The standard term to be used for this parameter is: **`DEDSL_VERSION`**

Occurrence If the module identification information is provided, then this parameter is mandatory and can appear only once.

Convention for the parameter value The format of this parameter value is a Dewy number represented in ASCII, with a maximum length of 30 characters.

Example parameter value **`0.1`**

3.3.2 DEDSL Module Identification: MODULE_TITLE

Purpose This parameter indicates a human readable name for the DEDSL module.

Term to be used The standard term to be used for this parameter is: **`MODULE_TITLE`**

Occurrence This parameter is optional and can appear only once.

Convention for the parameter value For this parameter the following conventions must be followed for its value:

- it shall be a free format character string that can span a number of lines;
- it may include any number of white space characters;
- the maximum length of the value of this parameter is 400 characters.

Example parameter value **`impQ_test dictionary`**

3.3.3 DEDSL Module ADID: MODULE_ADID

Purpose	This parameter indicates the CCSDS Authority and Description Identifier (ADID) that the DEDSL module has been registered under at a Control Authority (see references [9] and [10]).
Term to be used	The standard term to be used for this parameter is: **`MODULE_ADID`**
Occurrence	This parameter is optional and can appear only once.
Convention for the parameter value	The format of this parameter value is an unquoted ASCII string of eight consecutive Restricted ASCII [3] characters that constitute a registered MACAO ADID (see references [9] and [10]).
Example parameter value	**`NSSD1393`**

3.4 Implementation Guidelines

This section gives some guidelines concerning the use of attributes, whether 'standard', 'user defined' or 'identification' ones.

- The identifier for all attributes (independent if standard, user defined or module identification attributes) should follow the following guidelines:
  - Whilst the identifier is case sensitive, that is, the identifiers **`UNITS`**, **`Units`** and **`units`** are all different attribute identifiers (only **`UNITS`** being a 'standard' definition), it is **STRONGLY** recommended that users do not discern different attributes purely by case;
  - It is recommended that all identifiers only use uppercase letters, numerics and the underscore character (_).
- Whilst for any single semantic entity definition, the standard attributes can be presented or accessed in any order, it is recommended that the following order is used whenever possible so as to present a common style to all users (the optional attributes are indicated in italics):

  `NAME`
  `MEANING`
  `SHORT_MEANING`
  `UNITS`
  `ALIAS`
  `SPECIFIC_INSTANCE`
  `COMMENT`

  user defined attributes should follow the standard attributes.
- Whilst for any single user defined attribute definition, the parameters that define the user defined attribute can be presented or accessed in any order, it is recommended that the following order is used whenever possible so as to present a common style to all users (the optional attributes are indicated in italics):

[3] *For a definition of the characters that constitue the Restricted ASCII (RA) character set see Annex C.*

ATTRIBUTE_NAME
ATTRIBUTE_MEANING
ATTRIBUTE_VALUE_SYNTAX
ATTRIBUTE_EXTERNAL
ATTRIBUTE_OCCURANCE
ATTRIBUTE_VALUE_EXAMPLE
ATTRIBUTE_COMMENT

- The attributes for each entity definition (or parameters for each attribute definition) must be grouped in some manner so as to keep them separate from the attributes of other entity definitions. The methodology for grouping the attributes must be defined formally in the implementation syntax .
- When specifying the definition of entities that are described syntactically by a separate syntax language description, it is recommended to use the commonly understood 'dot' notation, for naming the data entities within the DEDSL implementation, that are part of a hierarchical data structure. For example, **`packet_header.time.minutes`**. Annex A.1 demonstrates this naming convention when using the syntax description language EAST (see References [1] and [2]).

4 IMPLEMENTATION USING PVL

Section 3 defines the DEDSL as an abstract standard, this permits the actual conveyance of the information via a number of techniques which may be necessary due to the manner that the data entities being described are actually formatted. The recommended method of conveying the information is by using the CCSDS developed Parameter Value Language (PVL, see references [5] and [6]).

PVL is designed to support the conveyance of named values, therefore it is ideal for the purpose of implementing the abstract standard defined in Section 3. The additional advantage of using PVL for the implementation is that it permits the inclusion of white-space characters and comments to assist layout and human readership. This makes the PVL implementation easily expandable and flexible.

If the data entity definitions or user defined attribute definitions are carried separately to the data entities being described, then each definition must be grouped into a collection of statements using the PVL statement aggregation techniques.

The following sections specify the implementation of the abstract standard in the order:

- Section 4.1 provides a structure diagram of a complete DEDSL module. This assumes that the definition is wholly implemented in PVL and is carried in a separate data block to the actual data.
- Section 4.2.1 defines the PVL implementation of the individual standard attributes.
- Section 4.2.2 defines the PVL implementation of how to group the standard attributes for each entity definition.
- Section 4.3.1 defines the PVL implementation of how to specify the parameters which define a user defined attribute.
- Section 4.3.2 defines the PVL implementation of how to group the parameters required to specify each user defined attribute.
- Section 4.4.1 defines the PVL implementation of the module identification attributes.
- Section 4.4.2 defines the PVL implementation of how to group the module identification attributes.

4.1 Complete DEDSL Definition

Figure 4-1 shows the structure of a complete DEDSL module. If the module identification attributes, user defined attributes and data entity definitions appear in a single data block, such as a file, then they must appear in the order indicated in this structure diagram. This is so as to present a consistent view to the user and for more efficient software processing. The contents of each of the three block are shown in Figure 4-2: Structure Diagram of a Single DEDSL Entity Definition , Figure 4-3: Structure Diagram of a Single DEDSL User Defined Attribute Definition and Figure 4-4: Structure Diagram of the DEDSL Module Identification Attributes . The structure diagram clearly shows which block of PVL statements are optional (off the centre line) and which are mandatory (on the centre line), those on the left of the centre line can appear at most only once and those on the right of the centre line can appear any number of times.

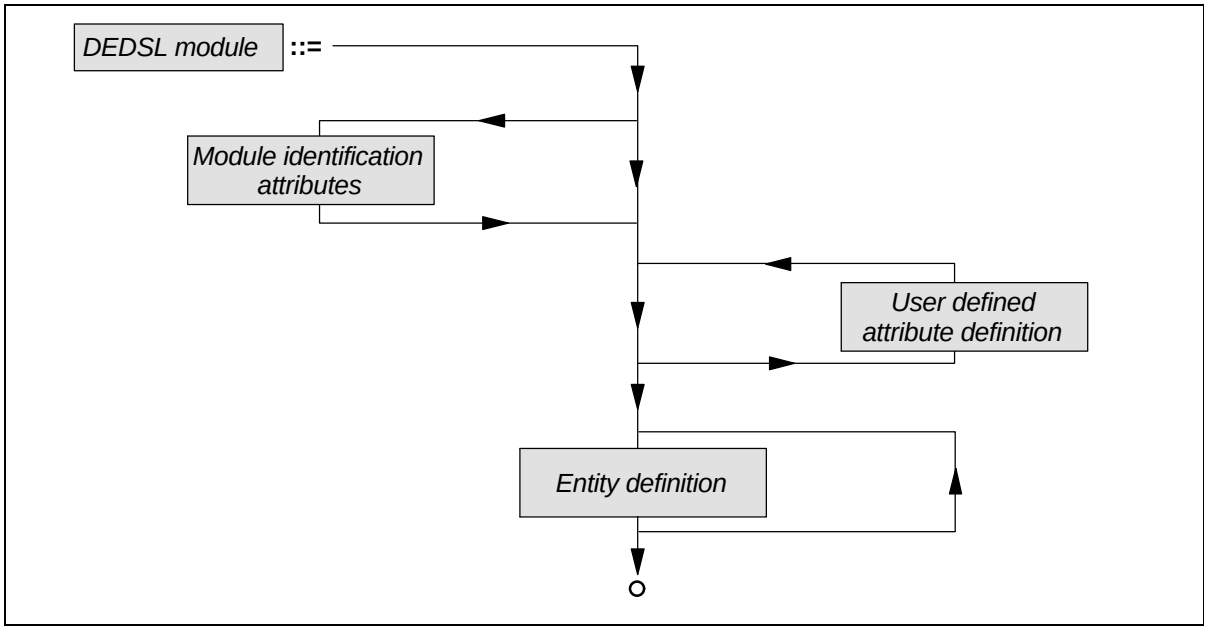


Figure 4-1: Structure Diagram of a Complete DEDSL Description

- Firstly appears the module identification attributes; whilst these are optional it is recommended that they should appear to assist in more efficient software support and configuration control.
- Secondly there are the user defined attribute definitions. There can be any number of these statement blocks.
- Thirdly, there are the entity definitions. There must be at least one of these within a DEDSL module and there is no limit on the total number.

A complete example using the recommended structure as defined in Figure 4-1 is shown at the end of Section 2 on page 9.

4.2 Standard Attributes

4.2.1 Implementation of Standard Attributes

Abstract Attribute	Data entity identification: NAME
PVL Parameter	**NAME**
PVL Value Syntax	An unquoted PVL string of maximum length 400 characters
Attribute occurrence	**1..1**
Example	**NAME = Epoch ;** **NAME = RMS ;**

Abstract Attribute	Full textual description of a data entity: MEANING
PVL Parameter	**MEANING**
PVL Value Syntax	A quoted PVL string of maximum length 8000 characters
Attribute occurrence	**1..1**
Example	**MEANING = “Time of mid-points of the observation intervals since A.D.” ;**

Abstract Attribute	Short textual description of a data entity: SHORT_MEANING
PVL Parameter	**SHORT_MEANING**
PVL Value Syntax	A quoted PVL string of maximum length 80 characters
Attribute occurrence	**0..1**
Example	**SHORT_MEANING = “Time line” ;**

Abstract Attribute	Scientific units of a data entity: UNITS
PVL Parameter	**UNITS**
PVL Value Syntax	A quoted PVL string of maximum length 80 characters
Attribute occurrence	**0..1**
Example	**UNITS = “0.225 kilo m/s2” ;** **UNITS = “ms” ;**

Abstract Attribute	Alternate identification of a data entity: ALIAS
PVL Parameter	**ALIAS**
PVL Value Syntax	A two value PVL sequence, the first value being an unquoted PVL string, of maximum length 400 characters, which is the alternate identifier for the data entity and the second value a quoted PVL string of maximum length 400 characters that explains the context for the use of the alternate identifier

Attribute occurrence	**0..n**
Example	**ALIAS = (SC_ID , "Short form for the spacecraft body identifier used for compactness within the spacecraft control system software") ;** **ALIAS = (TIME_LINE, "This entity is identified by this name within the ESA ground segement, rather than the primary name within the NASA ground segment") ;**

Abstract Attribute	Specific meaning of an instance of a data entity: SPECIFIC_INSTANCE
PVL Parameter	**SPECIFIC_INSTANCE**
PVL Value Syntax	A two value PVL sequence: the first value being an ASCII representation (i.e. for a binary number, the ASCII representation is as defined in reference [8]) of the specific instance of the data entity that has a specific meaning, this shall be no longer than 80 characters; the second value a quoted PVL string of maximum length 80 characters that explains the meaning of the specific instance.
Attribute occurrence	**0..n**
Example	**SPECIFIC_INSTANCE = (0, "Equator") ;** **SPECIFIC_INSTANCE = (100, "Boiling point of h-2-O") ;** **SPECIFIC_INSTANCE = (19920812T17:54:23.12, "Experiment switch on time") ;**

Abstract Attribute	Associated information about a data entity: COMMENT
PVL Parameter	**COMMENT**
PVL Value Syntax	A quoted PVL string of maximum length 8000 characters
Attribute occurrence	**0..n**
Example	**MEANING = " This entity definition is based upon that defined by the Planetary Data System developed by NASA/JPL. For further details about the PDS please contact JPL directly" ;**

4.2.2 Aggregation of a Data Entity Definition

Section 4.2.1 has specified the exact PVL syntax for each of the standard attributes required to define a single data entity. In the situation where more than one entity definition are conveyed in a single DEDSL module, such as a file, it is necessary to group all the attributes that describe a single entity, so as to distinguish each of the entity definitions. Using PVL, this is done with the GROUP construct.

Preceding the first of the grouped PVL statements, there shall be the PVL start aggregation statement:

```
BEGIN_GROUP = ENTITY_DEFINITION ;
```

Following the last of the grouped statements, there shall be the PVL end aggregation statement:

```
END_GROUP = ENTITY_DEFINITION ;
```

Figure 4-2 shows with a structure diagram the structure of the PVL statements that constitute a DEDSL entity definition. Whilst the order of the statements are not mandated to be in the order shown, this is the recommended order to enhance standardisation and understanding. The structure diagram clearly shows which PVL statements are optional (off the centre line) and which are mandatory (on the centre line), those on the left of the centre line can appear at most only once and those on the right of the centre line can appear any number of times.

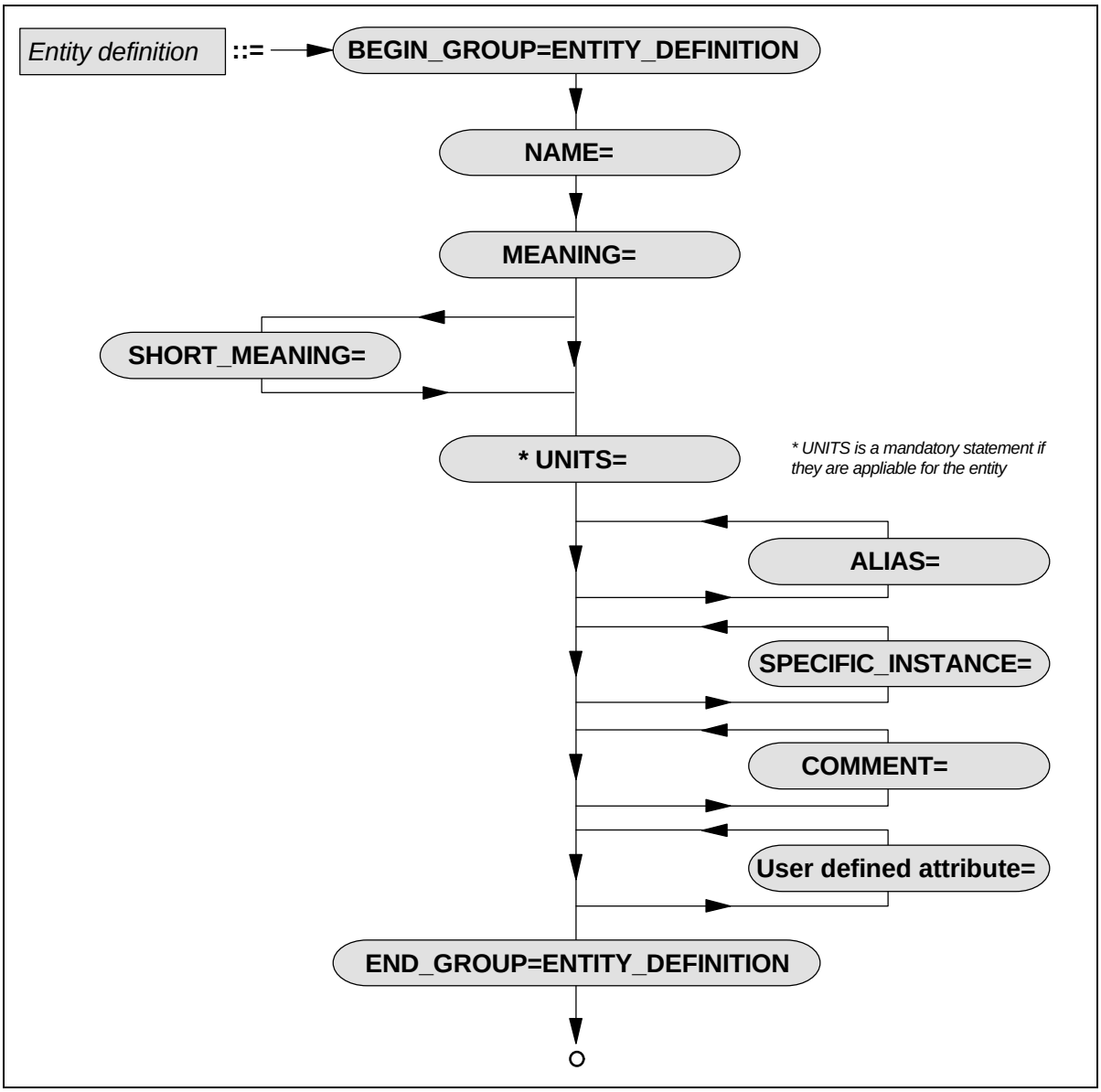


Figure 4-2: Structure Diagram of a Single DEDSL Entity Definition

Using the recommended structure above, the following shows how two entity definitions would appear in a single file (note in the second entity definition some of the standard attributes are not applicable and therefore not included):

```
BEGIN_GROUP = ENTITY_DEFINITION;
    NAME = Epoch ;
    MEANING = "Time of mid-points of the observation intervals since
               A.D." ;
    SHORT_MEANING = “Time Line" ;
    UNITS = ms ;
    ALIAS = (Samle_Time, “The name of the entity used by the central
             processing algorithm” );
    SPECIFIC_INSTANCE = ( 19900212T08:17:24.04, “Start of mission” );
    COMMENT = “This is an example entity to demonstare the DEDSL” ;
END_GROUP = ENTITY_DEFINITION;
```

```
BEGIN_GROUP = ENTITY_DEFINITION ;
    NAME = RMS ;
    MEANING = "Components of B in GSE coordinates obtained by taking
               the root-mean-square of the values in the observing
               intervals" ;
    SHORT_MEANING = "Components of RMS of B (GSE)" ;
    UNITS = “n tesla” ;
END_GROUP = ENTITY_DEFINITION;
```

4.3 User Defined Attributes

4.3.1 Implementation of User Defined Attributes

Abstract Parameter	Identification of an attribute: ATTRIBUTE_NAME
PVL Parameter	**ATTRIBUTE_NAME**
PVL Value Syntax	An unquoted PVL string of maximum length 40 characters
Parameter occurrence	**1..1**
Example	**ATTRIBUTE_NAME = PROJECT ;**

Abstract Parameter	Full textual description of an attribute: ATTRIBUTE_MEANING
PVL Parameter	**ATTRIBUTE_MEANING**
PVL Value Syntax	A quoted PVL string of maximum length 8000 characters
Parameter occurrence	**1..1**
Example	**ATTRIBUTE_MEANING = "This attribute specifies the name of the project for which the data entity attribute has been defined" ;**

Abstract Parameter	Syntax of an attribute value: ATTRIBUTE_VALUE_SYNTAX
PVL Parameter	**ATTRIBUTE_VALUE_SYNTAX**
PVL Value Syntax	One of the following unquoted PVL strings: **NUMERIC**, **INTEGER, REAL, STRING, LITERAL, TIME**, **SEQUENCE** or **EXTERNAL**. When the user defined attribute is used in an entity definition then the following PVL statement types shall be used to represent the value: **Parameter Value PVL Statement Type (see reference [R5])**

	NUMERIC : 'numeric' **INTEGER** : 'integer' or 'non decimal integer representations' **REAL** : 'floating point' **STRING** : 'quoted string' **LITERAL** : 'unquoted string' **TIME:** PVL time/date format **ENUMERATION:** 'unquoted string' **SEQUENCE** : 'sequence' **EXTERNAL** : 'quoted string'
Parameter occurrence	**1..1**
Example	**ATTRIBUTE_VALUE_SYNTAX = STRING ;** **ATTRIBUTE_VALUE_SYNTAX = SEQUENCE ;**

Abstract Parameter	Syntax of an attribute value: ATTRIBUTE_EXTERNAL
PVL Parameter	**ATTRIBUTE_EXTERNAL**
PVL Value Syntax	A PVL unquoted string consisting of 8 Restricted ASCII characters [4]
Parameter occurrence	**0..1**
Example	**ATTRIBUTE_EXTERNAL = EESA1234 ;**

Abstract Parameter	Example of the value of an attribute: ATTRIBUTE_VALUE_EXAMPLE
PVL Parameter	**ATTRIBUTE_VALUE_EXAMPLE**
PVL Value Syntax	The value of this parameter shall be the ASCII representation of the syntax that is defined by the parameter **ATTRIBUTE_VALUE_ SYNTAX** parameter
Parameter occurrence	**0..n**
Example	**ATTRIBUTE_VALUE_EXAMPLE = ISTP ;**

Abstract Parameter	Associated information about an attribute:

[4] *For a definition of the characters that constitue the Restricted ASCII (RA) character set see Annex C.*

	ATTRIBUTE_COMMENT
PVL Parameter	**ATTRIBUTE_COMMENT**
PVL Value Syntax	A quoted PVL string of maximum length 8000 characters
Parameter occurrence	**0..n**
Example	**ATTRIBUTE_COMMENT = "A formal list of recognised project names can be found out by contacting J. Smith, within NASA/NSSDC." ;**

4.3.2 Aggregation of User Defined Attribute Parameters

Section 4.3.1 has specified the exact PVL syntax for each of the parameters required to define a single user defined attribute. In the situation where more than one user defined attribute definition is conveyed in a single DEDSL module, such as a file, it is necessary to group all the parameters that describe a single user defined attribute, so as to distinguish each of the separate definitions. Using PVL, this is done with the GROUP contract.

Preceding the first of the grouped PVL statements, there shall be the PVL start aggregation statement:

BEGIN_GROUP = ATTRIBUTE_DEFINITION ;

Following the last of the grouped statements, there shall be the PVL end aggregation statement:

END_GROUP = ATTRIBUTE_DEFINITION ;

Figure 4-3 shows with a structure diagram the structure of the PVL statements that constitute a DEDSL user defined attribute definition. Whilst the order of the statements are not mandated to be in the order shown, this is the recommended order to enhanced standardisation and understanding. The structure diagram clearly shows which PVL statements are optional (off the centre line) and which are mandatory (on the centre line), those on the right of the centre line can appear any number of times.

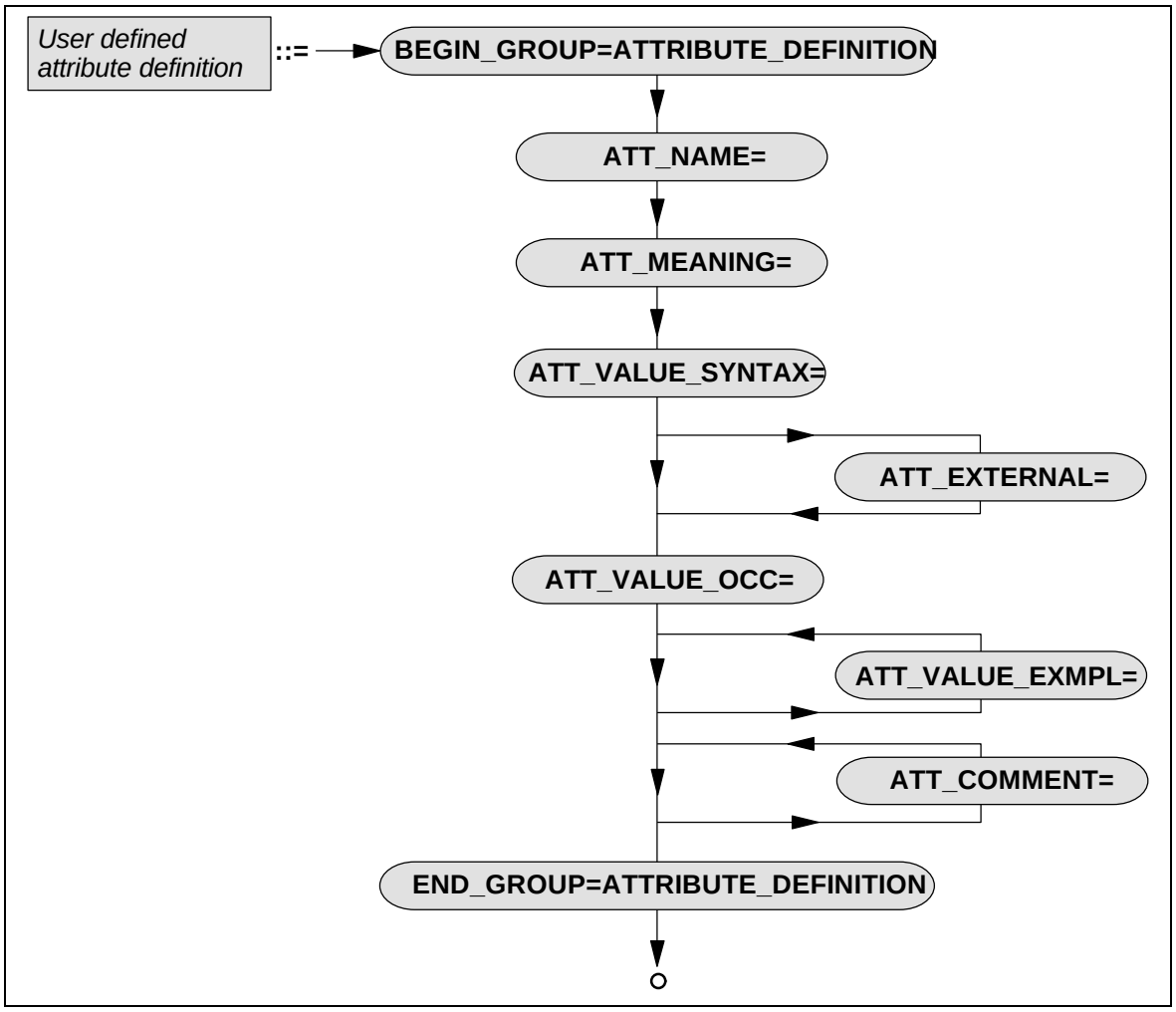


Figure 4-3: Structure Diagram of a Single DEDSL User Defined Attribute Definition

Using the recommended structure above, the following shows how two user defined attribute definitions would appear in a single file:

```
BEGIN_GROUP = ATTRIBUTE_DEFINITION ;
    ATTRIBUTE_NAME = PROJECT ;
    ATTRIBUTE_MEANING = "This attribute specifies the name of the project
                  for which the data entity attribute has been defined";
    ATTRIBUTE_VALUE_SYNTAX= STRING ;
    ATTRIBUTE_OCCURANCE = 0..1 ;
    ATTRIBUTE_VALUE_EXAMPLE= ISTP ;
    ATTRIBUTE_COMMENT = "A formal list of recognised project names can be
                         found out by contacting J. Smith, within
                         NASA/NSSDC." ;
END_GROUP = ATTRIBUTE_DEFINITION ;

BEGIN_GROUP = ATTRIBUTE_DEFINITION ;
    ATTRIBUTE_NAME = VALIDMAX ;
    ATTRIBUTE_MEANING = "This is a single scalar value that indicates
                         the maximum value that the entity may be
                         within any single instance of the data
                         structure." ;
    ATTRIBUTE_VALUE_SYNTAX = TIME ;
    ATTRIBUTE_OCCURANCE = 1..1 ;
END_GROUP = ATTRIBUTE_DEFINITION ;
```

4.4 DEDSL Module Identification Attributes

4.4.1 Implementation of DEDSL Module Identification Attributes

Abstract Attribute	Version of the DEDSL recommendation used: DEDSL_VERSION
PVL Parameter	**DEDSL_VERSION**
PVL Value Syntax	An unquoted PVL string of maximum length 30 characters representing a Dewy number
Attribute occurrence	**1..1**
Example	**DEDSL_VERSION = 0.1 ;**

Abstract Attribute	Identification of the DEDSL module: MODULE_TITLE
PVL Parameter	**MODULE_TITLE**
PVL Value Syntax	A quoted PVL string of maximum 400 characters
Attribute occurrence	**0..1**
Example	**MODULE_TITLE = "impQ_test dictionary" ;**

Abstract Attribute	MACAO ADID that the DEDSL module is registered under: MODULE_ADID
PVL Parameter	**MODULE_ADID**

PVL Value Syntax	A PVL unquoted string consisting of 8 Restricted ASCII characters [5]
Attribute occurrence	**0..1**
Example	**MODULE_ADID = NSSD1393;**

4.4.2 Aggregation of DEDSL Module Identification Attributes

Section 4.4.1 has specified the exact PVL syntax for each of the DEDSL module identification attributes. In the situation where all the module identification attributes are conveyed in a single data block, such as a file, it is necessary to group them, so as to distinguish them from the actual entity definitions. Using PVL, this is done with the GROUP construct.

Preceding the first of the grouped PVL statements, there shall be the PVL start aggregation statement:

BEGIN_GROUP = MODULE_IDENTIFICATION ;

Following the last of the grouped stamens, there shall be the PVL end aggregation statement:

END_GROUP = MODULE_IDENTIFICATION ;

Figure 4-4 shows with a structure diagram the structure of the PVL statements that constitute the DEDSL module identification attributes. Whilst the order of the statements are not mandated to be in the order shown, this is the recommended order to enhanced standardisation and understanding. The structure diagram clearly shows which PVL statements are optional (off the centre line) and which are mandatory (on the centre line), those on the left of the centre line can appear only once.

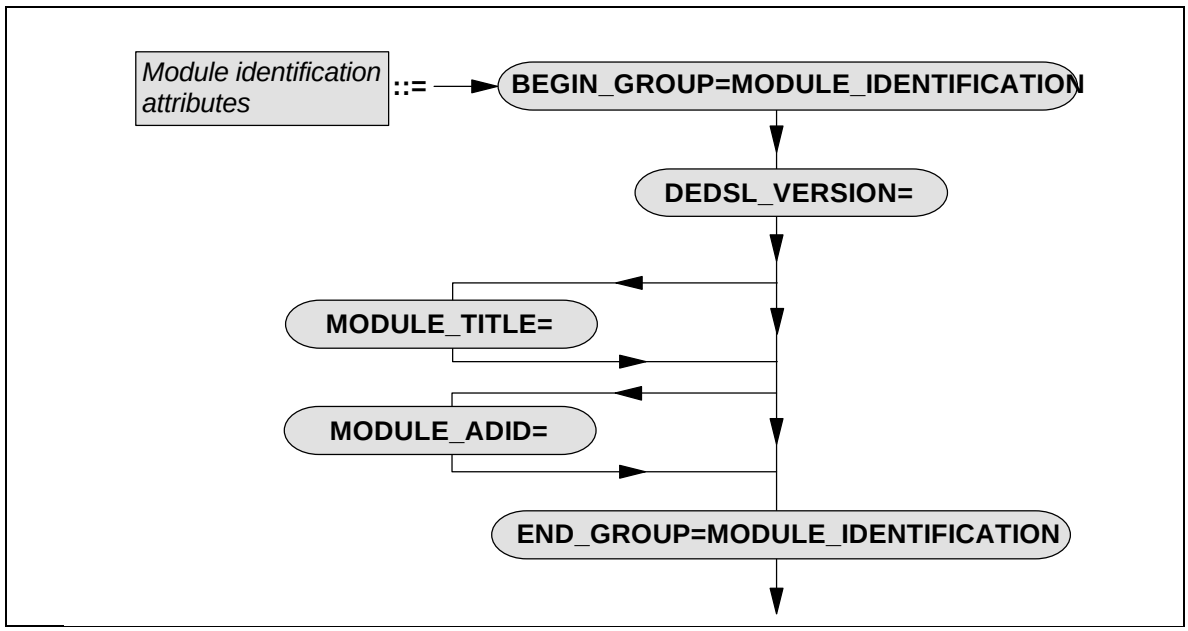


Figure 4-4: Structure Diagram of the DEDSL Module Identification Attributes

[5] *For a definition of the characters that constitue the Restricted ASCII (RA) character set see Annex C.*

Using the recommended structure above, the following figure shows an example of DEDSL module identification attributes:

```
BEGIN_GROUP = MODULE_IDENTIFICATION ;
    DEDSL_VERSION = 0.1 ;
    MODULE_TITLE = “impQ_test dictionary” ;
    MODULE_ADID = NSSD1393 ;
END_GROUP = MODULE_IDENTIFICATION ;
```

5 DEDSL CONFORMANCE

The assignment of unique identifiers, in the form of CCSDS ADIDs, is needed for the DEDSL material in order to facilitate the automated interpretation of data objects conforming to the DEDSL.

This DEDSL specification, is organised to permit two primary ways of using the specification. Section 3 ('ABSTRACT STANDARD') defines a set of standard attributes by name with restrictions on their permitted values. Annexes B and C provide further detailed restrictions on the attribute values. Note that this part of the specification does not specify how the attribute names and values are to be linked to any given data object. This allows a variety of formatting approaches to be used for this linking and this is the first primary way of using the DEDSL. For example, the CDF format and supporting software already provides attribute-value linking. CDF data objects could be written conforming to the specifications of Section 3. Such CDF data objects could be SFDU encapsulated and thus described by a registered description. This registered description could indicate that the CDF data objects were constrained to conform to DEDSL Section 3. This would be done by referencing the ADID assigned to DEDSL Section 3 within the registered description.

The second primary way of using the DEDSL is to use the material of Section 4 ('IMPLEMENTATION USING PVL') with that of the previous section. As an example of usage, consider a data object with an associated registered description. The description could contain an LVO with Class ID = E (meaning semantic description data), which in turn, contains a set of attributes and values that apply to numerous instances of the data object type. If the attributes in the LVO with Class ID = E conformed to the DEDSL specification of both Sections 3 and 4, then it would carry the ADID assigned jointly to both Sections 3 and 4 of this Recommendation.

Therefore, to support automated recognition of data conforming to the DEDSL, it is necessary to assign two unique identifiers or ADIDs. This results in two levels of DEDSL conformance.

5.1 Conformance Level 1: Abstract DEDSL (ADID = ZCSD0011)

A data object, recognised as being described by an ADID of **ZCSD0011**, shall be in conformance with the following sections of this Recommendation:

Section 1, Section 2, Section 3, Annex B and Annex C

5.2 Conformance Level 2: Complete DEDSL (ADID = ZCSD0012)

A data object, recognised as being described by an ADID of **ZCSD0012**, shall be in conformance with the following sections of this Recommendation:

Section 1, Section 2, Section 3, Section 4, Annex B and Annex C

Annex A: SCENARIOS OF DEDSL USAGE

(This annex **is not** part of the Recommendation)

Annex A.1: Using the DEDSL with a Formal Data Description Language

This scenario describes the process of creating the semantic description for a data structure which has been described using the formal syntax data description language EAST (see references [1] and [2]). This scenario assumes that the syntax description has been produced prior to the creation of the formal semantic description information.

This scenario uses the EAST example in Section 3.4.1 (pages 3-66 to 3-69) of the "EAST Tutorial (CCSDS 644.0-W-04)" (see reference [2]) called **`logical_CNES_description_01`** as the sample formal syntax description. The following is the stages that would be followed so as to produce the semantic description of the data:

- The data producer uses an interpreter of the EAST language to provide the list of fully qualified names of the data entities in the product. This includes the names of the base type entities and also the names of the higher level aggregated data structures. The names are:

```
TIME
TIME.Experiment_year
TIME.Experiment_month
TIME.Experiment_day
TIME.Experiment_hour
TIME.Experiment_minute
TIME.Experiment_second
BULLETIN_AT_THAT_TIME
BULLETIN_AT_THAT_TIME.Kind
BULLETIN_AT_THAT_TIME.State_Position_Element_X_Axis
BULLETIN_AT_THAT_TIME.State_Position_Element_Y_Axis
BULLETIN_AT_THAT_TIME.State_Position_Element_Z_Axis
BULLETIN_AT_THAT_TIME.State_Velocity_Element_X_Axis
BULLETIN_AT_THAT_TIME.State_Velocity_Element_Y_Axis
BULLETIN_AT_THAT_TIME.State_Velocity_Element_Z_Axis
BULLETIN_AT_THAT_TIME.Semi_Major_Axis
BULLETIN_AT_THAT_TIME.Eccentricity
BULLETIN_AT_THAT_TIME.Inclination
BULLETIN_AT_THAT_TIME.Right_Ascencion_Ascending_Node
BULLETIN_AT_THAT_TIME.Argument_of_Perigee
BULLETIN_AT_THAT_TIME.True_anomalie
INTERVAL
BULLETIN_AFTER_INTERVAL
BULLETIN_AFTER_INTERVAL.Kind
BULLETIN_AFTER_INTERVAL.State_Position_Element_X_Axis
BULLETIN_AFTER_INTERVAL.State_Position_Element_Y_Axis
BULLETIN_AFTER_INTERVAL.State_Position_Element_Z_Axis
```

```
BULLETIN_AFTER_INTERVAL.State_Velocity_Element_X_Axis
BULLETIN_AFTER_INTERVAL.State_Velocity_Element_Y_Axis
BULLETIN_AFTER_INTERVAL.State_Velocity_Element_Z_Axis
BULLETIN_AFTER_INTERVAL.Semi_Major_Axis
BULLETIN_AFTER_INTERVAL.Eccentricity
BULLETIN_AFTER_INTERVAL.Inclination
BULLETIN_AFTER_INTERVAL.Right_Ascencion_Ascending_Node
BULLETIN_AFTER_INTERVAL.Argument_of_Perigee
BULLETIN_AFTER_INTERVAL.True_anomalie
```

- The data producer creates a DEDSL module using a word processor, or a DEDSL creation utility if available, which takes as input the ASCII file of entity names produced by the EAST interpreter. A few examples of these DEDSL entity definitions are shown below. One decision the data producer must make is what data objects are relevant from a semantic viewpoint and therefore must have data entity definitions provided.

```
BEGIN_GROUP = ENTITY_DEFINITION ;
   NAME     = TIME;
   MEANING  ="The date of the current BULLETIN_AT_THAT_TIME";
END_GROUP   = ENTITY_DEFINITION ;

BEGIN_GROUP = ENTITY_DEFINITION ;
   NAME     = KIND;
   MEANING  ="The way the current bulletin is expressed (cartesian
             i.e. position vector and speed vector, or keplerian i.e.
             orbit description and position on this orbit)";
   COMMENT  ="If KIND is set to CARTESIAN then a cartesian bulletin is
             provided. If KIND is set to KEPLERIAN then a keplerian
             bulletin is provided.";
END_GROUP   = ENTITY_DEFINITION ;

BEGIN_GROUP = ENTITY_DEFINITION ;
   NAME     = INTERVAL ;
   MEANING  ="The number of seconds to be added to TIME to obtain the
             time of the BULLETIN_AFTER_INTERVAL";
   UNITS    ="second";
END_GROUP   = ENTITY_DEFINITION ;

BEGIN_GROUP = ENTITY_DEFINITION ;
   NAME     = BULLETIN_AT_THAT_TIME.State_Position_Element_X_Axis;
   MEANING  ="First component of the following vector (origin : Earth
             center, extremity : spacecraft gravity center)";
   UNITS    ="meter";
   ALIAS    = (X, “The identifier used with the processing software”);
END_GROUP   = ENTITY_DEFINITION ;

BEGIN_GROUP = ENTITY_DEFINITION ;
   NAME     = BULLETIN_AT_THAT_TIME.State_Velocity_Element_X_Axis;
   MEANING  ="First component of the vector modelising the velocity of
             the spacecraft.";
   UNITS    ="meter/second";
   ALIAS    = (V_X, , “The identifier used with the processing
                       software”);
```

```
END_GROUP   = ENTITY_DEFINITION ;

BEGIN_GROUP = ENTITY_DEFINITION ;
   NAME     = BULLETIN_AFTER_INTERVAL.Inclination;
   MEANING  ="Angle between the orbit plan and the plan orthogonal to
             the Earth rotation axis";
   UNITS    ="radian";
   ALIAS    = (I, "The identifier used with the processing software");
END_GROUP   = ENTITY_DEFINITION ;
```

- The data producer registers the syntax description written in EAST and the semantic description written in DEDSL (**ZCCSD0012**), at a Control Authority Office (see references [9] and [10]) and is assigned an ADID which can then be attached to all the application data.

Annex A.2: The DEDSL with a Simple CDF Example

In this scenario, we will attempt to use the data representation capabilities of the Common Data Format (CDF) to carry both the data entities of interest and the DEDSL attribute information specified for conformance to level 1 (see Section 5.1). Note that a CDF models its data as containing a set of global attributes describing the overall CDF content (i.e. the CDF entity) and a number of attributes for each variable (which is also an entity) within the CDF structure (i.e., CDF entity value).

A project has chosen the Common Data Format (CDF) as the file format and access mechanism that it will use to carry most of its primary data of interest. The project has not standardised on the names and meanings of any of the global attributes or the individual 'variable' entity attributes to be used. A simple CDF, consisting of six global attributes and two 'variable' entities with seven attributes per entity , is created. The schematic in Figure 2-1 on page 9 is a close representation of this CDF, but it has a few less attributes for clarity . A structured textual overview of the CDF file content is shown below:

```
impQ_test

Global attributes:
    NAME            CDF_CHAR    "impQ_test"
    MEANING         CDF_CHAR    "This CDF contains magnetic field
                                 observations from Interplanetary
                                 Monitoring Platform Q (IMP-Q) as
                                 prepared for testing.  There will
                                 probably be multiple instances of CDFs
                                 having the name CDF_impQ_test."
    SHORT_MEANING   CDF_CHAR    "IMP-Q Magnetic Field"
    PROJECT         CDF_CHAR    "International Solar-Terrestrial Physics"
    TEXT            CDF_CHAR    "Data: 1.024 minute averages"
    ADID_REF        CDF_CHAR    "NSSD1393"

Variables and their attributes:
    Epoch           CDF_EPOCH
        NAME            CDF_CHAR     "Epoch"
        MEANING         CDF_CHAR     "Time of mid-points of the
                                      observation intervals since A.D."
```

```
        SHORT_MEANING    CDF_CHAR     "Time Line"
        UNITS            CDF_CHAR     "m second"
        VALIDMIN         CDF_EPOCH    "01-Jan-1990 00:00:00.000"
        VALIDMAX         CDF_EPOCH    "31-Dec-2010 23:59:59.000"
        FILLVAL          CDF_REAL8    "-10.0e30"

    RMS              CDF_REAL4
        NAME             CDF_CHAR     "RMS"
        MEANING          CDF_CHAR     "Components of B in GSE
                                       coordinates obtained by taking
                                       the root-mean-square of the
                                       values in the observing
                                       intervals"
       SHORT_MEANING     CDF_CHAR     "Components of RMS of B (GSE)"
       UNITS             CDF_CHAR     "n tesla"
       TVALIDMIN         CDF_REAL4    "0.0, 0.0, 0.0"
       TVALIDMAX         CDF_REAL4    "20.0, 20.0, 20.0"
       FILLVAL           CDF_REAL8    "-10.0E30"
```

- For each global attribute (i.e. attributes of the complete 'CDF entity') and each variable attribute (i.e., attribute of a 'variable' entity), the attribute name, attribute value syntax and attribute value is specified.
- For each 'variable' entity, the entity name and entity value syntax is specified. The value of each 'variable' entity consists of a large number of simple data type values and therefore it is not shown for clarity.

The standard semantic names given in Section 3 have been used where appropriate and where they fit naturally within the constraints and expressiveness of the CDF syntax.

The following should be clear from this example:

1. For the global attributes of the CDF **impQ_test**, only **NAME**, **MEANING** and **SHORT_MEANING** apply from the standard attributes defined in Section 3.1. Therefore the global attributes **PROJECT**, **TEXT**, and **ADID_REF** would be understood as 'user defined attributes' from a DEDSL perspective. Here the 'user' is the project.
2. For the value of the CDF **impQ_test**, there are two entities called **Epoch** and **RMS**. The **Epoch** entity has several attributes. The standard attributes **NAME**, **MEANING,** **SHORT_MEANING** and **UNITS** are as defined in Section 3.1. The **UNITS** attribute, which defines the units for the value of the **Epoch** entity, is in '**m second**' (milli-seconds), which conforms with those specified in Annex B. The attributes **VALIDMIN**, **VALIDMAX**, and **FILLVAL** would be understood as 'user defined attributes.'
3. For the entity **RMS**, basically the same attributes apply as for **Epoch**. Note that the value of the entity **RMS** is a sequence of three real numbers, corresponding to the components of the magnetic field in three dimensions. Therefore the **TVALIDMAX** and **TVALIDMIN** attributes also have three components for their values.

This CDF is not fully conformant to DEDSL level 1 because the 'user defined attributes' do not have the required attribute definition information required by DEDSL Section 3. There is, currently, no CDF standard way to express such information, although it could be put into the text field of a new global attribute. However there would be no place to define this new global attribute.

In conclusion, it can be said that while it is not possible to put a current CDF instance, standing alone, into DEDSL level 1 conformance, nevertheless the degree of conformance shown above is beneficial and it is recommended that such a level of conformance be encouraged. As will be shown in the CDF example of Annex A.5, a CDF that is augmented by association with a DEDSL module can be put into full conformance with DEDSL level 2 which includes level 1 conformance.

Annex A.3: Using the DEDSL with FITS

This scenario describes the use of the DEDSL to improve the semantic definition and hence understanding of a Flexible Image Transport System (FITS) file taken from the ESA XMM mission.

The XMM Optical Monitor Periodic Housekeeping telemetry data corresponding to a specific exposure during a single observation is held in a FITS file. The contents of this FITS file is as follows:

Primary Header: containing general details of the exposure

```
SIMPLE  =                    T / Standard FITS format
BITPIX  =                    8 / Bytes
NAXIX   =                    0 / No data associated with Primary Header
EXTEND  =                    T / Extensions present
INSTRID =                   OM / Instrument identifier
OBSID   =                RF243 / Observation Identifier
EXPID   =                    4 / Exposure identifier
```

First Extension Header: containing the optical monitor housekeeping telemetry data which is sampled every 8 seconds

```
XTENSION= 'BINTABLE'         T   / Binary table extension
BITPIX  =                    8   / Bytes
NAXIS   =                    2   / 2D matrix
NAXIS1  =                   14   / Number of bytes per row
NAXIS2  =                   64   / Number of rows
PCOUNT  =                    0   / No data values precede the table
GCOUNT  =                    1   / One table
TFIELDS =                    6   / Number of fields in each row
EXTNAME = 'OM_HK_A '             / OM Periodic Housekeeping - Data Type A
UTCATIM =  19950812T03:42:35  / UTC time of the first data type A
structure
TFORM1  = 'C       '
TTYPE1  = 'DICH_POS'
TUNIT1  = '        '
TFORM2  = 'I       '
TTYPE2  = 'B1FW_POS'
TUNIT2  = 'degrees '
TFORM3  = 'I       '
TTYPE3  = 'B2FW_POS'
TUNIT3  = 'degrees '
TFORM4  = 'C       '
TTYPE4  = 'HVENAB_1'
TUNIT4  = '        '
TFORM5  = 'J       '
TTYPE5  = 'TCPC1024'
TUNIT5  = '        '
TFORM6  = 'J       '
```

```
TTYPE6  = 'TMPC1024'
TUNIT6  = '        '
END
¾$ ¿ý%¹ü ;ür ´LÍ!ýó¥ü‹÷¿  --‹è² éŽ$vis 4 Ò ÿ»éá ¸ LÍs!¸ ûÿ? d  ± ÓàŽÀ¿þÿò
 °Ãüò®ã ë ñ•ó øÃ&€=€uíúßG} úuæù òßÿþù suØù }ôuÑù ëÊù÷ß üuÐ} u¼ù¿• uduµùø
ù÷ïýu ù u™ù±•Âu'ù u‹ÿ èu„&Ç .ÿûïÃE .Aú kúwÞ î .EúÇø î  gùÃŠ>ÿq3 º Í ¾Ù¿Æ
 àñ,Ì‹MMAûUO»•Ä Í s 3ÀúNJVÿñ¾¬< t ´ Óëõ^ÿñÃ‹ófÆ ƒÇ ¹ëó¦tè@ àyà=ú » ‹ú ‹ ü
  ;çÓ^Ó tï'ï  ?¿  ¶ ¾Æßo‹Ls‹TOŠ µ{ßaºº=ïOuá¨~w ï=Ã×ïðç í ••?a7 þuã•j&u
úkƒu ú•Çl.u úm uún# u úo• Ð˜• àñ,Ì‹MMAûUO»•ÄÆ Í s 3ÀúNJVÿñ¾¬ ´Óëõ^ÿñÃ‹ófÆ
 ƒ ¹ëó¦tè@ àyà=ú »• ‹ ú ‹ ü ³ ;çÓ^Ó tï'ï ?¿  ¶ ¾Æßo‹Ls‹TOŠ µ{ßaºº=ïOuá¨
~w ï=Ã×ïðç í ••? þuã•j&u úkƒu ú•Çl.u úm uún# u úo• Ð˜•
```

In this FITS file, the information before the **END** statement describes the actual binary exposure data that follows the **END** statement. The FITS file contains not only standard FITS keywords, but also the keywords to define the binary table. The data itself is logically formatted as a table like so:

DICH_POS	B1FW_POS	B2FW_POS	HVENAB_1	TCP1024	TMPC1024

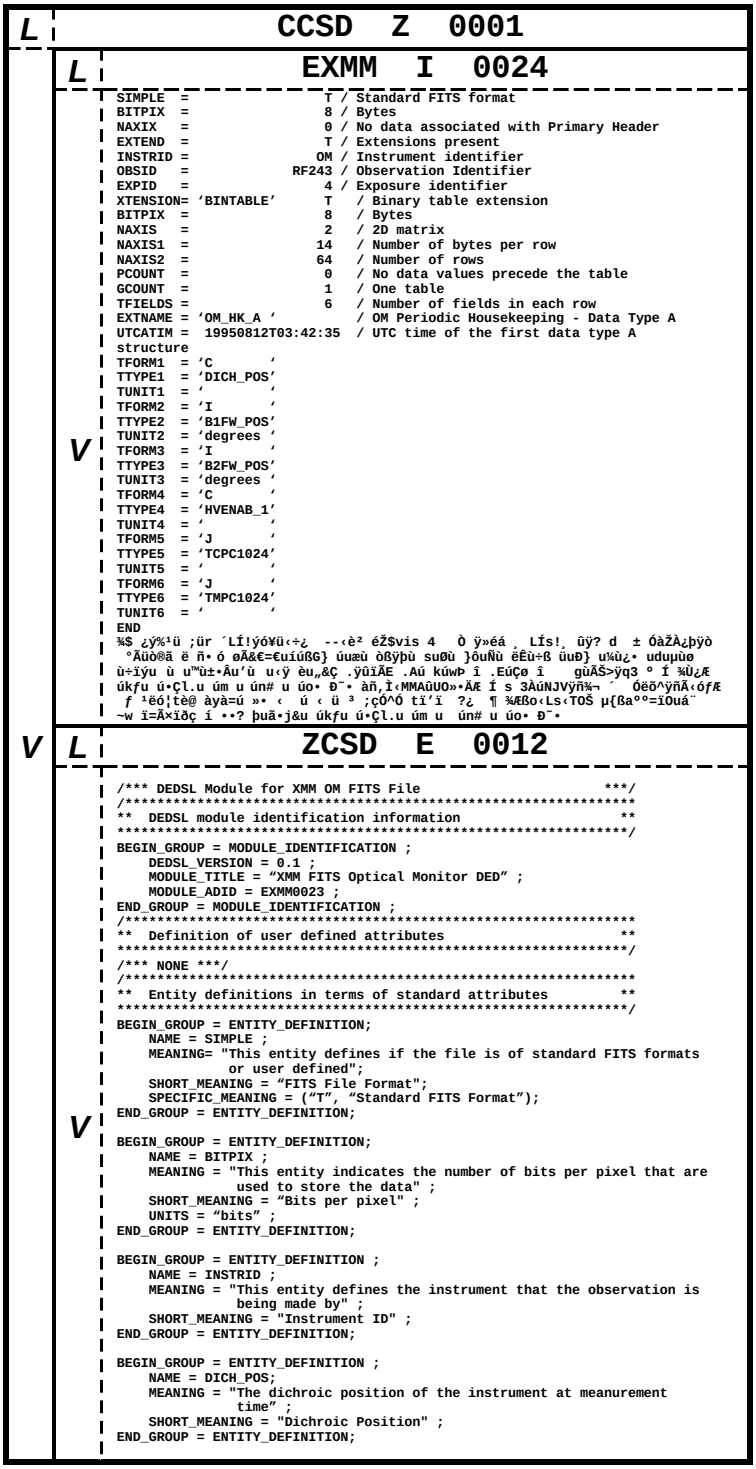
To define each of the table column headings (i.e. repeatable entities), the keywords **TFORMi, TTYPEi** and **TUNITi** are used. These keywords define the entity type, entity name and entity units respectively. This set of six entities is then repeated many times (corresponding to each eight second sample rate). Whilst there is the possibility to put small comments to the right of the FITS values as shown above, and the definitions of each entity is partly given by the **TTYPEi**, **TFORMi** and **TUNITi** statements, these are not satisfactory to unambiguously convey the full meaning of the data entities. To do this the DEDSL may be used. Furthermore, the standard FITS keyword are also not clearly defined to a new user - these can also be formally described by the DEDSL. Below is an example DEDSL module, demonstrating the definition of the data entities above (definitions for only a sample of the entities shown above are shown in the DEDSL example below):

```
/*** DEDSL Module for XMM OM FITS File                          ***/

/*************************************************************
**  DEDSL module identification information                   **
*************************************************************/

BEGIN_GROUP = MODULE_IDENTIFICATION ;
    DEDSL_VERSION = 0.1 ;
    MODULE_TITLE = "XMM FITS Optical Monitor DED" ;
    MODULE_ADID = EXMM0023 ;
END_GROUP = MODULE_IDENTIFICATION ;

/*************************************************************
**  Definition of user defined attributes                     **
*************************************************************/
```

```
/*** NONE ***/

/************************************************************
**  Entity definitions in terms of standard attributes        **
************************************************************/

BEGIN_GROUP = ENTITY_DEFINITION;
    NAME = SIMPLE ;
    MEANING= "This entity defines if the file is of standard FITS formats
              or user defined";
    SHORT_MEANING = “FITS File Format";
    SPECIFIC_MEANING = (“T”, “Standard FITS Format”);
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
    NAME = BITPIX ;
     MEANING = "This entity indicates the number of bits per pixel that
are
               used to store the data" ;
    SHORT_MEANING = “Bits per pixel" ;
    UNITS = “bits” ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION ;
    NAME = INSTRID ;
    MEANING = "This entity defines the instrument that the observation is
               being made by" ;
    SHORT_MEANING = "Instrument ID" ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION ;
    NAME = DICH_POS;
    MEANING = "The dichroic position of the instrument at meanurement
               time” ;
    SHORT_MEANING = "Dichroic Position" ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION ;
    NAME = B1FW_POS ;
    MEANING = "The position of the Blue1 filter wheel in relation to the
               normal of the exposure plane” ;
    SHORT_MEANING = "Blue1 fileter wheel position” ;
    UNITS = “degrees” ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION ;
    NAME = TCPC1024 ;
    MEANING = "A count of the number of telecommand packets received with
                     application identifer equal to 1024 and therefore
commanding
               the Blue1 filter wheel position” ;
    SHORT_MEANING = "TC Packet Count of APID=1024” ;
END_GROUP = ENTITY_DEFINITION;

......
```

```
/*****************************************************************
/**                End of data entity dictionary                **
*****************************************************************/
```

The data in the FITS file and the semantic definition in the DEDSL file could then be packaged together in an SFDU as shown schematically below, so as to convey the full meaning of the FITS data with the actual data itself, therefore leading to a greater probability of the data being of value in the future.



```
BEGIN_GROUP = ENTITY_DEFINITION ;
    NAME = B1FW_POS ;
    MEANING = "The position of the Blue1 filter wheel in relation to the
               normal of the exposure plane" ;
    SHORT_MEANING = "Blue1 fileter wheel position" ;
    UNITS = "degrees" ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION ;
    NAME = TCPC1024 ;
    MEANING = "A count of the number of telecommand packets received with
               application identifer equal to 1024 and therefore
commanding
               the Blue1 filter wheel position" ;
    SHORT_MEANING = "TC Packet Count of APID=1024" ;
END_GROUP = ENTITY_DEFINITION;
......
/*****************************************************************
/**              End of data entity dictionary                  **
******************************************************************/
```

Rather than package the Class E LVO with the FITS data each time, it would also be possible to register the DEDSL module with a Control Authority (see References [9] and [10]) and then only request it from the Control Authority just the once.

Annex A.4: Using the DEDSL with HDF

The Hierarchical Data Format (HDF) is designed for the interchange of raster image data and multi-dimensional scientific data sets across heterogeneous environments. It is a multi-object file format, with a number of predefined object types, such as arrays, but with the ability to extend the object types in a relatively simple manner. Recently HDF has been extended to handle arbitrary scientific data, rather than just uniform array oriented data, and also annotation attribute data.

HDF can store several types of data objects within one file, such as raster images, palettes, text and table style data. Each 'object' or 'entity' in a HDF file has a predefined tag that indicates the data type and a reference number that identifies the instance. A table of contents is maintained within the file and as the user adds data to the file the pointers in the table of contents are updated. An example organisational structure of a HDF file is shown in Figure A-1 .

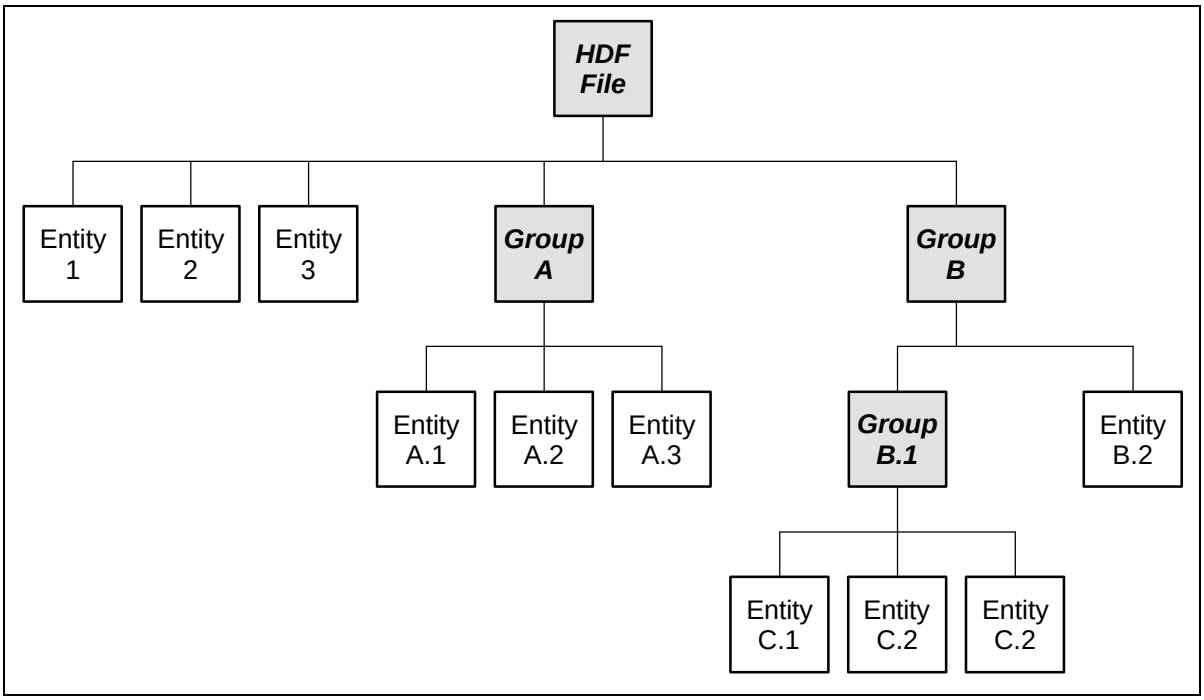


Figure A-1: Example Organisation of a HDF File

In this example, all the data within the single HDF product is stored within a single file. The physical format of the file in unknown to the user and the only means of accessing the data is through a software library. The physical storage is in a canonical form and therefore the files can be transferred to other platforms (assuming the software library is also available on the target platform).

HDF has the capability of adding attributes to either the complete file, the groups of entities or each entity within a group. There is no defined format or standard for these attributes they are purely textual strings. Therefore to use the DEDSL in collaboration with a HDF file, one would still use the PVL representation as presented in Section 4 of this Recommendation to define each of the semantic pieces of information, i.e. **MEANING**, **UNITS**, etc. but the PVL grouping statements (i.e., **BEGIN_GROUP=ENTITY_ DEFINITION**) would not be required as the DEDSL definition can be 'attached' to each object in the HDF as a single textual string. This is shown schematically in Figure A-2 (only a subset of the example shown in Figure A-1).

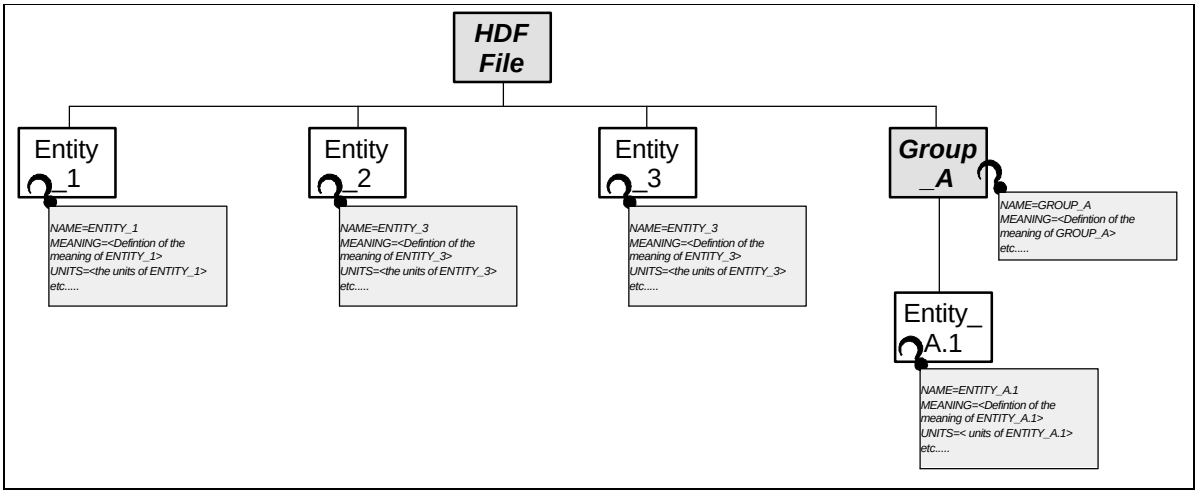


Figure A-2: DEDSL Usage with a HDF File

As can be seen the DEDSL Module Identification attributes are attached as a HDF annotation at the complete file level and then the entity definitions are attached to the separate objects within the HDF file at the object level. Each of the annotations are formatted in PVL as defined in Section 4.

Annex A.5: Using the DEDSL with CDF

In this scenario, we will examine the use of an external DEDSL module to add semantics, and to provide more standardisation in the semantics, associated with a Common Data Format (CDF) object (instance) or type. Doing this will improve the possibility of finding the CDF type carrying the data entities of interest, and it will improve opportunities for automated services dealing with multiple entities. Note that a CDF models its data as containing a set of global attributes describing the overall CDF content (i.e. the CDF entity) and a number of named variables, each with its own set of named 'variable attributes.' In Annex A.2, the entity model in Section 2 was mapped to the CDF model by relating CDF attributes to various DEDSL attributes. For this scenario we use a different,

and more flexible, mapping. Here we treat each named data value in CDF as a data entity. This allows each named CDF data value (be it a CDF global attribute, CDF variable, or CDF variable attribute) to be further annotated with DEDSL attributes in an external DEDSL module. These DEDSL attributes can carry additional, and more standardi sed, semantics.

A project has chosen the CDF as the file format and access mechanism that it will use to carry most of its data of interest. The project has standardised on the names and meanings of many of the global attributes and the variable attributes to be used but not on the names or meanings of the variables themselves. A simple CDF, holding data from a particular magnetometer instrument, might appear conceptually as follows:

```
CDF NAME:  imp9_test
Dimensions & Sizes:  1,3

Global attributes:
        Title           CDF_CHAR    (IMP-9 Magnetic Field)
        Project         CDF_CHAR    (International Solar-Terrestrial bs.)
        TEXT            CDF_CHAR    (Data: 1.024 minute averages)
        ADID_ref        CDF_CHAR    (NSSD1393)

Variables and their attributes:
        Epoch           CDF_EPOCH   Record: T, Dimension: F
             FIELDNAME          CDF_CHAR     (Time Line)
             VALIDMIN           CDF_EPOCH    (01-Jan-1990 OO:00:00.000)
             VALIDMAX           CDF_EPOCH    (31-Dec-2010 23:59:59.000)
             UNITS              CDF_CHAR     (ms)
             FILLVAL            CDF_REAL8    (-10.0e30)
             VAR_TYPE           CDF_CHAR     (data)

        RMS             CDF_REAL4   Record: T, Dimension: T
             FIELDNAME          CDF_CHAR     (Component of RMS of B (GSE))
             VALIDMIN           CDF_REAL4    (0.0, 0.0, 0.0)
             VALIDMAX           CDF_REAL4    (20.0, 20.0, 20.0)
             UNITS              CDF_CHAR     (nT)
             FILLVAL            CDF_REAL8    (-10.0E30)
             VAR_TYPE           CDF_CHAR     (data)
             LABL_PTR_1         CDF_CHAR     (label_rms)

        RMS_p           CDF_REAL4   Record: T, Dimension: F
             FIELDNAME          CDF_CHAR     (Magnetic Field RMS)
             VALIDMIN           CDF_REAL4    (0.0)
             VALIDMAX           CDF_REAL4    (40.0)
             UNITS              CDF_CHAR     (nT)
             FILLVAL            CDF_REAL8    (-10.0e30)
             VAR_TYPE           CDF_CHAR     (data)

        label_rms       CDF_CHAR    Record: F, Dimension: T
             FIELDNAME          CDF_CHAR     (Labels for RMS of B (GSE))
             VAR_TYPE           CDF_CHAR     (metadata)

[1] = "Bx_RMS (GSE)"
[2] = "By_RMS (GSE)"
[3] = "Bz_RMS (GSE)"
```

The **Dimensions & Sizes** of **1,3** means that this CDF recognises a variation in one dimension in addition to the variation from record to record, and that one dimension has three components (values). It can be seen that each attribute consists of a name, a data type for its value (e.g., **CDF_CHAR**), and the actual value of the attribute (in parenthesis). By contrast, each variable (except **label_rms**) is shown as a name and data type but not with a value. For **Epoch**, **RMS**, and **RMS_p**, the amount of data is too great to show. However for **label_rms**, there are only three values and these are shown in the lower left corner of the listing . For each variable, its variation along the record axis and along the single dimension axis is also indicated by **TRUE(T)** or **FALSE(F)**.

A typical CDF file of this type would have many records where each would have an **Epoch** time together with three components of magnetic field and the magnetic field magnitude. The label information for the three component of magnetic field (**label_rms**), whose actual values are shown above, would be in a separate record.

The global attribute **ADID_ref** provides an internationally unique 8-character identifier, called an ADID, which is given as **NSSD1393** for this particular CDF type. This identifies a description of this CDF type. Such a description will be called a CDF type description in the remainder of this scenario. This CDF type description has been registered with an agency's CCSDS Control Authority (see references [9] and [10]) and may be obtained upon request. Note that, for each CDF type, there are likely to be multiple CDF files, all with the same CDF_NAME of '**imp9_test**' and the same named variables and attributes.

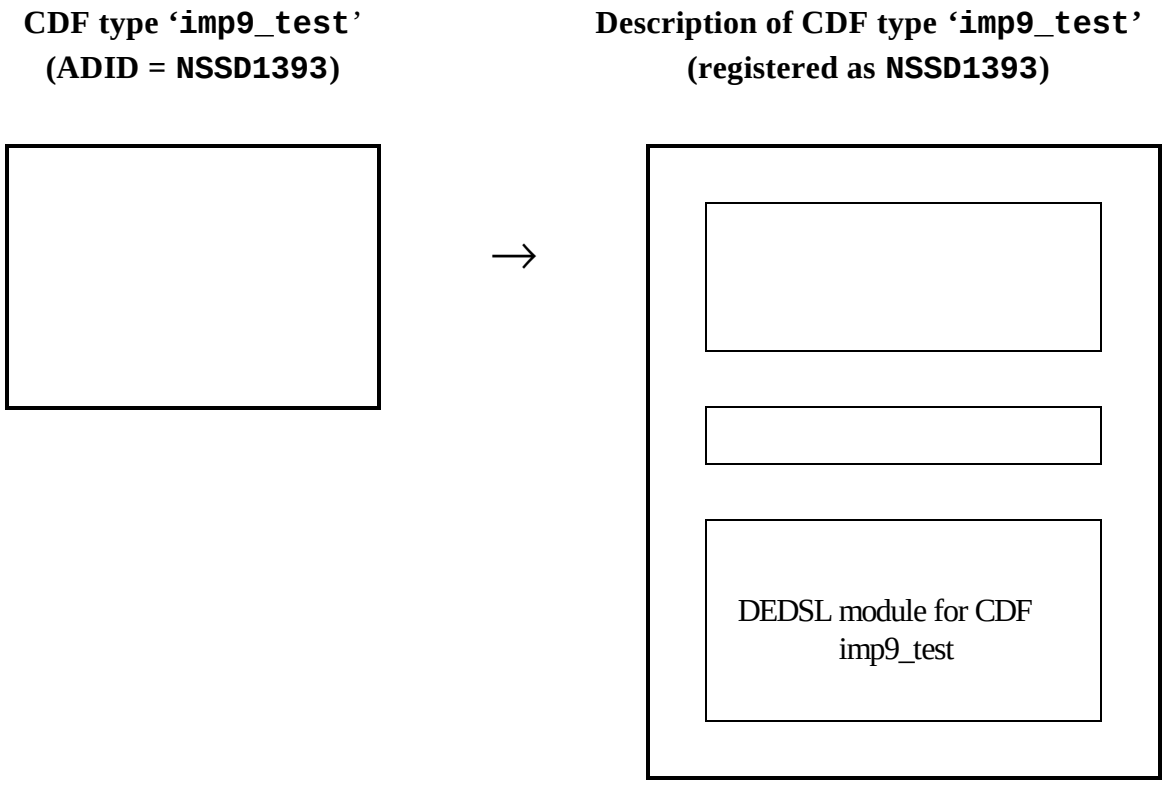
In our scenario many project CDF types and instances (files), from different instruments, have been generated. The producer of each CDF type has used the same set of attribute names shown above, where applicable. However there is little consistency in CDF variable names nor in the **FIELDNAME** attribute values to help other users of the data find and use the variables of interest. Many additional CDFs (including new CDF types) will be generated in the future. The project wishes to associate, with each CDF type, a standard set of semantics to make it much easier for others to find and use the variables desired. Also, it wants to extend the readily available semantics for some of the CDF attributes and variables.

The most practical approach, which does not require rewriting existing CDFs, is to associate an externally recorded set of standard semantics with each instance of a CDF. If desired, such semantics could be incorporated directly into new CDFs with the inclusion of new attributes. However this may not be the best approach in terms of efficient data searching and management.

The project adopts a standard set of 'science' and 'supporting' class words and their meanings to be used to categorise the meaning of the CDF variables of primary interest to investigators. Sets of sub-class words, with each set used to modify one of the class words, are also adopted. These words and definitions, together with the constraints on their relationships, comprise the primary content of one of the project data dictionaries. Coupling this data dictionary with a syntax and grammar for expressing simple combinations of these words results in a language (which is also data type) that can be used to describe, in a more standard way, the meaning of CDF variables typically used in space physics investigations. To make this language widely visible and permanently available, the project registers this language with an agency's Control Authority (see references [9] and [10]) and receives an ADID (i.e., **NSSD2001**) to use as a standard reference identifier for this language (or data type).

Since the CDF type descriptions have already been registered (e.g., **NSSD1393** in the example above), each CDF instance already has an externally recorded and available set of semantics that applies to all the instances of that CDF type. Therefore the project approach is to add a DEDSL module to the existing registered description of each CDF type, and to include in the DEDSL module the names of the CDF variables for which additional semantics will be added and the DEDSL attributes that will carry this additional semantics. Also included in the DEDSL module will be the names of CDF attributes for which

additional semantics are desired. Note that the CDF DEDSL module for each CDF type will generally be different because each CDF type has its own set of variable names and possibly unique, required, semantic extensions. Schematically, using the CDF type example shown above, this looks as follows:



Generally, there will be many instances of CDF type **imp9_test**, but each is described by the one registered description **NSSD1393**. This description has several description objects within it, but the only one of interest here is the one in the form of a DEDSL module as specified by this Recommendation.

The DEDSL module for CDF **imp9_test** is shown in the following figure:

```
/******                    begin dictionary preface                ******/

BEGIN_GROUP = MODULE_IDENTIFICATION;
    DEDSL_VERSION = 0.1;
    MODULE_TITLE = "imp9_test DEDSL Dictionary";
    MODULE_ADID = NSSD1393;
END_GROUP = MODULE_IDENTIFICATION;

BEGIN_GROUP = ATTRIBUTE_DEFINITION;
    ATTRIBUTE_NAME = STANDARD_MEANING;
    ATTRIBUTE_MEANING = "This attribute's value carries a standard set of
                         semantics expressed in the project language/
                         data-type defined in the description NSSD2001";
    ATTRIBUTE_VALUE_SYNTAX = EXTERNAL;
    ATTRIBUTE_EXTERNAL = NSSD2001;
    ATTRIBUTE_OCCURANCE = 0..1;
```

```
END_GROUP = ATTRIBUTE_DEFINITION;

/******              begin dictionary entity list             ******/

BEGIN_GROUP=ENTITY_DEFINITION;
     NAME=imp9_test;
     MEANING= "This CDF was generated from data recorded every 16
              milliseconds and typically 64 observations are included in
              each average, yielding an average time interval between
              values of 1.024 seconds";
END_GROUP=ENTITY_DEFINITION;

BEGIN_GROUP=ENTITY_DEFINITION;
    NAME = ADID_ref;
    MEANING = ”This 8 character identifier is internationally unique and
               identifies a description of this CDF type (last 4
               characters) and the organization (first 4 characters) that
               holds the description";
END_GROUP=ENTITY_DEFINITION;

BEGIN_GROUP=ENTITY_DEFINITION;
    NAME = Epoch;
    MEANING =  “This variable carries the mid-point time of observations
                referenced from AD”;
    UNITS = “m second”;
    STANDARD_MEANING = "time>Epoch_center_range";
END_GROUP=ENTITY_DEFINITION;

BEGIN_GROUP=ENTITY_DEFINITION;
    NAME = RMS;
    MEANING = “Provides the root-mean-square of the cartesian
               components in the GSE coordinate system”;
    UNITS = “n tesla”;
    STANDARD_MEANING = "magnetic_field>GSE_cartesian_vector";
END_GROUP=ENTITY_DEFINITION;

BEGIN_GROUP=ENTITY_DEFINITION;
    NAME = RMS_p;
    MEANING = “Provides the root-mean-square of the RMS variable values”;
    UNITS = “n tesla”;
    STANDARD_MEANING = "magnetic_field>average_amplitude";
END_GROUP=ENTITY_DEFINITION;

BEGIN_GROUP=ENTITY_DEFINITION;
    NAME =  RMS.VALIDMAX;
    MEANING = “The value of 20 nano tesla is obtained when the
               instrument goes into saturation mode.  The instrument
               also saturates when the electron gun fires, but in these
               cases we have inserted  the fill value (see FILLVAL)”;
END_GROUP=ENTITY_DEFINITION;

/*                      end of dictionary                       */
```

The DEDSL module begins with a **MODULE_IDENTIFICATION** section. The **MODULE_ADID** attribute indicates that this DEDSL module has been registered in the description identified as **NSSD1393**. Decoding this ADID provides information on how to

contact the CCSDS Control Authority office known as '**NSSD**' that holds this description as '**1393**' (see References [9] and [10]).

The next section of the DEDSL module, the **ATTRIBUTE_DEFINITION** section, defines a 'user-defined' attribute called **STANDARD_MEANING**. A text description of the meaning is given. It states that the description registered as **NSSD2001** defines a language (or data type) that will be used to form the value of the **STANDARD_MEANING** attribute. This is represented formally by the **ATTRIBUTE_VALUE_SYNTAX** and **ATTRIBUTE_EXTERNAL** statements. The **ATTRIBUTE_VALUE_SYNTAX** statement states that the syntax of the **STANDARD_MEANING** attribute's value will be identified by the **ATTRIBUTE_EXTERNAL** statement. The **ATTRIBUTE_EXTERNAL** statement states that the description **NSSD2001** should be used to understand the content of the **STANDARD_MEANING** attribute's string. Note also that the **STANDARD_MEANING** attribute is not required to be present in every entity definition within the DEDSL module. However if it is present, it must only appear once (**ATTRIBUTE_OCCURRENCE = 0..1**).

In the **ENTITY_DEFINITION** section of the DEDSL module, six of the entities in the CDF are selected to have their semantics enhanced. The first entity, which is the entire CDF named **imp9_test**, is given a text description using the DEDSL **MEANING** standard attribute. There is no single understanding of units for this entity, so no **UNITS** attribute is required.

The second entity, called **ADID_ref**, is a CDF global attribute. It's understanding is updated with a text description using the DEDSL module **MEANING** standard attribute.

The third entity, called **Epoch**, is a standard CDF variable that is well documented in the CDF documentation. Here it's understanding is made more directly accessible, and it is given a standard set of searchable semantics through the use of the user-defined attribute **STANDARD_MEANING**. The value of this attribute, shown as **time>Epoch_center_range**, is a language statement that conforms to the language defined by the description registered as **NSSD2001**. **Time** is an allowed class word and the words following the '**>**' are modifier sub-class words.

The fourth entity, called **RMS**, is a CDF variable. It was picked by the data product designer and is not standardised outside a narrow community. Therefore the additional DEDSL semantics provided by the **STANDARD_MEANING** attribute gives greater understanding and should enhance finding by interested users.

The fifth entity, called **RMSp**, is similar to **RMS**.

The last entity in the DEDSL **ENTITY_DEFINITION** section, called **RMS.VALIDMAX**, is a CDF RMS-variable attribute. Its name is given using the 'dot' notation to distinguish it from all the other **VALIDMAX** attributes in the CDF. The semantics for this CDF attribute are expanded to discuss two cases in which a maximum value might have been thought to occur. If the entity name had been given as **VALIDMAX**, then this additional semantics would have applied to all entities whose name matched **VALIDMAX**.

By adding similar DEDSL modules to the other descriptions registered for the CDF types, the project is able to enhance the amount and uniformity of the semantics of the CDF collection without having to re-write any of the CDFs. Further, the collection of DEDSL

modules can be separately accessed and searched when looking for data entities of interest.

Annex B: Permitted Unit Names and Prefixes

(This annex **is** part of the Recommendation)

This Annex shows the configuration file (**`udunits.dat`**) to be used with the NCAR Unidata units processing software. This file is set up to only accept a subset of the units that the complete package will accept so as to encourage standardisation of units across data sets. The permitted units are listed in the left hand column of the listing.

Any of the units may be multiplied by one of the factors in the following table, this is done by preceding the unit with the string indicated in the 'Representation of UNITS Attribute' column:

Prefix	**Factor by which the unit is multiplied**	**Internationally Recognised Scientific Symbol**	**Representation in `UNITS` Attribute**
exa	10^{18}	E	**`E`**
peta	10^{15}	P	**`P`**
tera	10^{12}	T	**`T`**
giga	10^{9}	G	**`G`**
mega	10^{6}	M	**`M`**
kilo	10^{3}	k	**`k`**
hecto	10^{2}	h	**`h`**
deca	10^{1}	da	**`da`**
deci	10^{-1}	d	**`d`**
centi	10^{-2}	c	**`c`**
milli	10^{-3}	m	**`m`**
micro	10^{-6}	m	**`u`**
nano	10^{-9}	n	**`n`**
pico	10^{-12}	p	**`p`**
femto	10^{-15}	f	**`f`**
atto	10^{-18}	a	**`a`**

Table B-1: Representation of Unit Prefixes

```
# udunits.dat
#
# The first column is the unit name as should be used as the value for a
# UNITS attribute.
# The second column indicates whether or not the unit name has a plural
```

```
# form (i.e. can have an 's' appended).
# A 'P' indicates that the unit has a plural form, whereas, a 'S' indicates
# that the unit has a singular form only.  The remainder of the line is the
# definition for the unit.
#
# Anything after the '#', to-end-of-line is a comment.
#

#
# BASE UNITS.  These must be first and are identified by a nil definition.
#
ampere                  P                       # electric current
bit                     P                       # unit of information
candela                 P                       # luminous intensity
kelvin                  P                       # thermodynamic temperature
kilogram                P                       # mass
meter                   P                       # length
mole                    P                       # amount of substance
second                  P                       # time
radian                  P                       # plane angle

#
# CONSTANTS
#
percent                 S 0.01
PI                      S 3.14159265358979323846

#
# NB: All subsequent definitions must be given in terms of
# earlier definitions.  Forward referencing is not permitted.
#

#
# The following are non-base units of the fundamental quantities
#

#
# UNITS OF ELECTRIC CURRENT
#
gilbert                 P 7.957747e-1 ampere
statampere              P 3.335640e-10 ampere

#
# UNITS OF THERMODYNAMIC TEMPERATURE
#
celsius                 S kelvin @ 273.15
rankine                 P kelvin/1.8
fahrenheit              P rankine @ 459.67

#
# UNITS OF MASS
#
carat                   P 2e-4 kilogram
gram                    P 1e-3 kilogram         # exact
kg                      S kilogram
g                       S kg/1000
atomic_mass_unit        P 1.66044e-27 kilogram
tonne                   P 1e3 kilogram          # exact
```

```
#
# UNITS OF LENGTH
#
m                       S meter
metre                   P meter
light_year              P 9.46055e15 meter
micron                  P 1e-6 meter            # exact
parsec                  P 3.085678e16 meter
au                      S 1.495979e11 meter

#
# UNITS OF AMOUNT OF SUBSTANCE
#
mol                     S mole

#
# UNITS OF TIME
#
day                     P 8.64e4 second         # exact
hour                    P 3.6e3 second          # exact
minute                  P 60 second             # exact
year                    P 3.153600e7 second     # exact

#
# The following are derived units with special names.  They are useful for
# defining other derived units.
#
steradian               P radian2
hz                      S 1/second
newton                  P kilogram.meter/second2
coulomb                 P ampere.second
lumen                   P candela steradian
becquerel               P 1/second              # SI unit of activity of a
#                                               # radionuclide
pascal                  P newton/meter2
joule                   P newton.meter
lux                     S lumen/meter2
watt                    P joule/second
volt                    P watt/ampere
farad                   P coulomb/volt
ohm                     P volt/ampere
siemens                 S ampere/volt
weber                   P volt.second
tesla                   P weber/meter2
henry                   P weber/ampere

#
# The following are compound units: units whose definitions consist
# of two or more base units.  They may now be defined in terms of the
# preceding units.
#
e                       S 1.6021917e-19 coulomb # charge of electron
eV                      P 1.60219e-19 joule
VA                      S volt ampere
hp                      S 7.456999e2 watt

#
```

```
# PRESSURE OR STRESS
#
bar                     P 1e5 pascal            # exact
at                      S 1 kg gravity/cm2

#
# RADIATION UNITS
#
curie                   P 3.7e10 becquerel      # exact
roentgen                P 2.58e-4 coulomb/kg    # exact

#
# VELOCITY (INCLUDES SPEED)
#
c                       S 2.997925e+8 meter/sec

#
# VISCOSITY
#
poise                   S 1e-1 pascal second    # absolute viscosity.
stokes                  S 1e-4 meter2/second    # exact
rhe                     S 10/(pascal second)    # exact

#
# VOLUME (INCLUDES CAPACITY)
#
cc                      S cm3
liter                   P 1e-3 m3             # exact. However, from 1901
                                              # to 1964, 1 liter =
                                              # 1.000028 dm3

#
# COMPUTERS AND COMMUNICATION
#
baud                    S 1/second              # exact
bps                     S bit/second

#
# MISC
#
rpm                     S hz/60
```

Annex C: ASCII & Restricted ASCII

(This annex **is** part of the Recommendation)

A code is a correspondence between a symbol and a number of digits of a number system. The American Standard Code for Information Interchange (ASCII) is a seven-bit code also known as the USA Standard Code for Information Interchange (USASCII). The latest updated American National Standards Institute ANSI-X3 standard for this is ANSI X3.4-1977. This code has been incorporated into the ISO code of the same nature (ISO 646-1983) which includes other symbols and alphabets. Since the ISO code is an eight-bit code, the ASCII code is embedded in an eight-bit field in which the higher order bit is set to zero. The Restricted ASCII set of characters (denoted here by the shaded codes with the ⇒ symbol pointing to them) is in this Recommendation. The primary reference to be used should be ISO 646-1983.

The ASCII and Restricted ASCII or RA codes are given in Table C -1. (The code for each character (Char) is given in decimal (Dec), and hexadecimal (Hex)).

Char	Dec	Hex		Char	Dec	Hex		Char	Dec	Hex		Char	Dec	Hex
NUL	0	00		space	32	20		@	64	40		`	96	60
SOH	1	01		!	33	21	⇒	**A**	**65**	**41**		a	97	61
STX	2	02		”	34	22	⇒	**B**	**66**	**42**		b	98	62
ETX	3	03		#	35	23	⇒	**C**	**67**	**43**		c	99	63
EOT	4	04		$	36	24	⇒	**D**	**68**	**44**		d	100	64
ENQ	5	05		%	37	25	⇒	**E**	**69**	**45**		e	101	65
ACK	6	06		&	38	26	⇒	**F**	**70**	**46**		f	102	66
BEL	7	07		’	39	27	⇒	**G**	**71**	**47**		g	103	67
BS	8	08		(	40	28	⇒	**H**	**72**	**48**		h	104	68
HT	9	09		)	41	29	⇒	**I**	**73**	**49**		i	105	69
LF	10	0A		*	42	2A	⇒	**J**	**74**	**4A**		j	106	6A
VT	11	0B		+	43	2B	⇒	**K**	**75**	**4B**		k	107	6B
FF	12	0C		,	44	2C	⇒	**L**	**76**	**4C**		l	108	6C
CR	13	0D		-	45	2D	⇒	**M**	**77**	**4D**		m	109	6D
SO	14	0E		.	46	2E	⇒	**N**	**78**	**4E**		n	110	6E
SI	15	0F		/	47	2F	⇒	**O**	**79**	**4F**		o	111	6F
DLE	16	10	⇒	**0**	**48**	**30**	⇒	**P**	**80**	**50**		p	112	70
DC1	17	11	⇒	**1**	**49**	**31**	⇒	**Q**	**81**	**51**		q	113	71
DC2	18	12	⇒	**2**	**50**	**32**	⇒	**R**	**82**	**52**		r	114	72
DC3	19	13	⇒	**3**	**51**	**33**	⇒	**S**	**83**	**53**		s	115	73
DC4	20	14	⇒	**4**	**52**	**34**	⇒	**T**	**84**	**54**		t	116	74
NAK	21	15	⇒	**5**	**53**	**35**	⇒	**U**	**85**	**55**		u	117	75
SYN	22	16	⇒	**6**	**54**	**36**	⇒	**V**	**86**	**56**		v	118	76
ETB	23	17	⇒	**7**	**55**	**37**	⇒	**W**	**87**	**57**		w	119	77
CAN	24	18	⇒	**8**	**56**	**38**	⇒	**X**	**88**	**58**		x	120	78
EM	25	19	⇒	**9**	**57**	**39**	⇒	**Y**	**89**	**59**		y	121	79

SUB	26	1A
ESC	27	1B
FS	28	1C
GS	29	1D
RS	30	1E
US	31	1F

:	58	3A
;	59	3B
<	60	3C
=	61	3D
>	62	3E
?	63	3F

⇒

Z	**90**	**5A**
[	91	5B
\	92	5C
]	93	5D
^	94	5E
_	95	5F

z	122	7A
{	123	7B
|	124	7C
}	125	7D
~	126	7E
DEL	127	7F

Table C-1: ASCII and Restricted ASCII Codes

Annex D: Sample User Defined Attributes

(This annex **is not** part of the Recommendation)

To be completed.